Software-Defined Networking Based Moving Target Defenses

A thesis

submitted in partial fulfillment

of the requirements for the Degree

of

Doctor of Philosophy

in the

University of Canterbury

by

Dilli Prasad Sharma

Department of Computer Science and Software Engineering

University of Canterbury

Christchurch, New Zealand

2020

Abstract

The static and homogeneous nature of the existing state-of-the-art networked systems provides asymmetric advantages to attackers that make them easy for reconnaissance and launching the attacks. The advanced cyberattacks (e.g., APT, DDoS, malware) cause tremendous socioeconomic impact and losses; therefore, there is an immediate need to not only respond to the already happened attacks but disrupt the attackers for preventing the attacks. The concept of moving target defense (MTD) is to dynamically change attack surface to increase uncertainty and confuse the attackers by invalidating the attacker's intelligence or information collected during the reconnaissance or procedures to launch the attacks. Software-defined networking (SDN) has emerged as a promising technology that provides flexibility and programmability to networked systems, which facilitates the implementation of the MTD operations for cybersecurity. MTD research based on SDN environments is in an infant stage since both MTD and SDN research areas have emerged relatively recently.

Identifying hosts and vulnerable services (e.g., IP addresses and TCP/UDP ports) of the target systems is a precursory step for the vast majority of the cyberattacks. Network address space randomization, random host mutation, and ports hopping are commonly used network address shuffling MTD approaches. However, there are problems in these existing MTD approaches. First, these MTD mechanisms randomize IP addresses or port numbers of a host mapping them to a virtual IP or port in a one-to-one manner, which requires more IP addresses to satisfy the mutation rate and

unpredictability constraints. Secondly, they lack to use both IP address and ports shuffling so that the services are vulnerable since they are long-time exposure to potential attackers. Thirdly, the existing security metrics are limited to particular attack scenarios or models, very application-specific, and lack to capture and measure the effectiveness of the various types of MTD techniques (e.g., shuffling of IP, ports, or network topology). This thesis aims to address the aforementioned problems in three primary research goals: (1) to develop a moving target defense mechanism that continuously and dynamically changes the virtual IP addresses of a server host in the SDN; (2) to design and develop a moving target defense mechanism that protects the SDN by shuffling both hosts' virtual IP addresses and services' port numbers in the SDN; and (3) to develop a set of dynamic security metrics for measuring the effectiveness of the SDN-based MTD techniques.

To achieve the goal (1), a new MTD approach Flexible Random Virtual IP Multiplexing, namely FRVM, is proposed, which aims to thwart network reconnaissance and scanning attacks in software-defined networks. FRVM enables a host machine to have multiple, random, time-varying, virtual IP addresses, which are multiplexed to a real IP address of the host. The FRVM frequently changes all the virtual IP addresses of the host over time; the multiplexing and de-multiplexing operations of it remaps the virtual IP to a real IP, and a real IP to the virtual IP address(s) of the host respectively. Therefore, at the end of every multiplexing event, the FRVM aims to make the attackers lose any knowledge gained through the reconnaissance and to disturb their scanning strategy. The performance evaluation of the FRVM is comparatively analyzed with a baseline model (i.e., a typical static network without FRVM mechanism) through simulations and experiments using the security (e.g., attacker success probability) and performance metrics (e.g., delay, throughput). The results show that the proposed FRVM effectively increases the attacker's work effort with acceptable operational overhead.

To achieve the goal (2), a new proactive, dynamic SDN-based MTD

mechanism Random Host and Service Multiplexing, namely RHSM, is developed. RHSM uses the multiplexing or de-multiplexing of both IP addresses (e.g., hosts) and port numbers (e.g., services) aiming to obfuscate both network and transport layers' real identities of the hosts and the services for defending against the network reconnaissance and scanning attacks. RHSM allows each host to use random, multiple virtual IP addresses to be dynamically and periodically shuffled. Also, it uses short-lived, multiple virtual port numbers for active services running on the host. The RHSM dynamically and frequently changes all the virtual IPs and the virtual ports of the host and services, respectively. The simulations and experimental results demonstrate the RHSM effectively outperform a baseline counterpart in terms of the attack success probability and defense cost.

To achieve the goal (3), new dynamic security metrics are developed. The proposed dynamic security metrics that timely, dynamically, and adaptively assess the effectiveness of the SDN-based MTD techniques. The security metrics are developed to measure the dynamics of a network and a host state's information (e.g., IP address, port, software stacks, vulnerabilities, or network topology) introduced by various types of MTD techniques shuffling them. The key aspect of our proposed metrics is to capture variability that keeps track of changing patterns of the network and the host states upon every MTD triggering event. The proposed metrics are following three categories: (i) Network and host address-based metrics that measure variability of the network and the host addresses based on a degree of uncertainty and unpredictability on the assigned IP address to the hosts in a network; (ii) Attack path-based metrics which are used to measuring variability of attack paths using graphical models estimated based on the network state transitions from one topology to another topology upon triggering a network topology and/or IP shuffling MTD; and (iii) attack stage-based success metrics measure the chances of discovering a vulnerable target host's information, exploiting the target host's vulnerability, and compromising the target host. Via extensive simulation

study, the key parameters that can significantly affect the performance of the MTD are investigated using the proposed security metrics.

In summary, the main contributions of this thesis are: (1) the development of the flexible random virtual IP multiplexing (FRVM) in the SDNs; (2) the evaluation of the security performance and overhead of the MTD in the SDNs; (3) the development of the random host and service multiplexing (RHSM) for MTD in the SDNs; and (4) the development of the dynamic security metrics for measuring effectiveness of the MTDs in the SDNs.

Dedicated to my beloved wife Brinda, and son Suhan.

Acknowledgement

I would like to express my sincere gratitude to everyone who supported me throughout my Ph.D journey. It would not have been possible to complete this research without their support and guidance.

First and foremost, I am incredibly grateful to my research Co-supervisor Dr. Dong Seong Kim, for his guidance, patience, motivation, and continuous support throughout the study. He has not only given me guidance and advice for my research work, but he has also supported me in all the aspects of my study and stay in New Zealand. I appreciate all the time he has spent in discussing my research work, motivating and encouraging me, and providing support for me to attend conferences. His guidance helped me in all the time of research and writing of this thesis. Thank you, and I will forever remain grateful for this tremendous support and generosity.

I am deeply indebted to my Senior Supervisor Dr. Andreas Willig and Associate Supervisor Dr. Walter Guttmann, for their advice, support, encouragement, and guidance. Thanks a lot for their precious time and support. Without their support, it would have been impossible for me to finish this thesis.

I also would like to express my sincere gratitude to Dr. Jin-Hee Cho (Virginia Tech, USA), Dr. Terrence J. Moore, Dr. Frederica F. Nelson (Army Research Lab, USA), Seunghyun Yoon, and Dr. Hyuk Lim (GIST, South Korea) for their invaluable comments, ideas, reviews, advice and suggestions during my research. Thank you all for their insightful comments and suggestions that allowed me to greatly improve the quality of the work

presented in this thesis.

My special gratitude goes to Dr. Jin B. Hong and Dr. Mengmeng Ge, for their invaluable advice, discussion, and feedback on my research. I am also indebted to my friend Dr. Simon Yusuf Enoch, for his invaluable insights and suggestions. I appreciate his willingness to help and meet me whenever I need it. Thank you all the members of the UC Cybersecurity Lab, Cole, Paul, Matthew, Sophie, Bilal, Abdul, Ke He, and Julio for interesting talks, discussions, and all the fun time we have had in the past years. Thank you to Dibash, Prerna, Andrew, Amelia, Daniel, Camila, Adam, Haipeng, Kashif and Sarmad for their cooperation.

Thank you to all the administrative staff of the Department for their instant help and support during my study. Thank you to Alex, Lynleigh, and Sharon for solving all my inquiries on conferences, administrative, and financial matters.

I gratefully appreciate all the funding sources the University of Canterbury- Department of Computer Science and Software Engineering Ph.D Scholarship, the G B Battersby-Trimble Scholarship, and the US Army Research Development and Engineering Command (RDECOM) International Technology Center-Pacific (ITC-PAC) and U.S. Army Research Laboratory (US-ARL) (through Dr. Dong Seong Kim) that made my Ph.D research possible.

Last but not least, I am grateful to my family, especially my wife Brinda, for her support and encouragement. This thesis would not have been possible without her patience and sacrifice. I also would like to thank all my relatives and friends for their support throughout the research journey.

Contents

List of Figures

List of Tables

List of Abbreviations

AG	Attack Graph
AP	Attack Path
APT	Advanced Persistent Threat
APV	Attack Path Variability
APVIS	Attack Path Variability with IP-Shuffling
ARP	Address Resolution Protocol
ASP	Attacker Success Probability
BS	Based Score
CKC	Cyber Kill Chain
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
CSP	Compromise Success Probability
CVE	Common Vulnerability and Exposure
CVSS	Common Vulnerability and Scoring System
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service

DSP	Defense Success Probability
ESP	Exploit Success Probability
FRVM	Flexible Random Virtual IP Multiplexing
GA	Genetic Algorithm
HARM	Hierarchical Attack Representation Model
HAV	Host Address Variability
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Thing
IP	Internet Protocol
IPS	Intrusion Prevention System
LAN	Local Area Network
MAC	Media Access Control
MAN	Metropolitan Area Network
ML	Machine Learning
MT6D	Moving Target Defense using IPv6
MTD	Moving Target Defense
MTTC	Mean Time To Compromise
MTTF	Mean Time To Failure
NAP	Number of Attack Paths
NASR	Network Address Space Randomization
NAV	Network Address Variability
NVD	National Vulnerability Database

OF	OpenFlow
OS	Operating System
RHM	Random Host Mutation
RHSM	Random Host and Service Multiplexing
SAPV	Shortest Attack Path Variability
SAPVIS	Shortest Attack Path Variability with IP-Shuffling
SCADA	Supervisory Control And Data Acquisition
SDN	Software-Defined Networking
SDR	Shuffling Diversity Redundancy
SSP	Scan Success Probability
TCP	Transmission Control Protocol
T-HARM	Temporal Hierarchical Attack Representation Model
UDP	User Datagram Protocol
VM	Virtual Machine
VNAP	Variability of the Number of Attack Paths

List of General Notations

h	is a host machine in a network
n	is the number of hosts in a network
N	is an available IP address space size
T	is a multiplexing (or MTD shuffling) interval time
θ	is a multiplexing (or shuffling) rate
Π	is a deterrence that measures the cost exerted on an attacker
Γ	is the effectiveness of deployed MTD mechanism
τ	is an overhead delay incurs due to deploying MTD
Δ	is an overhead delay percentage
Ω	is an overhead throughput percentage
ζ	is a mean length of flow-table
μ	is an average flow-update rate
$\mathbb{H}$	is a finite set of all hosts in a network
$\mathbb{RIP}$	is a finite set of all real IP addresses
$\mathbb{VIP}$	is a finite set of all virtual IP addresses
$\mathbb{P}$	is a finite set of all TCP/UDP port numbers
ASP_h	is an attacker success probability of discovering a host h in a static network

ASP_s	is an attacker success probability of discovering a service s in a static network
ASP_h^{rhsm}	is an attacker success probability of discovering a host h in a network with RHSM defense mechanism
ASP_s^{rhsm}	is an attacker success probability of discovering a service s in a network with RHSM defense mechanism
t_k	is a time point, where $k = 1, 2, 3, \ldots, m$
$\mathbb{NS}_{t_k}$	is a finite set of IP addresses used in a network at a time t_k
$\mathbb{HS}_{t_k}$	is a finite set of IP address currently assigned to a host at a time t_k
AG_{t_k}	is an attack graph at a time t_k
ap_{t_k}	is an attack path from the attacker to the target host at a time t_k
$\mathbb{AP}_{t_k}$	is a finite set of all attack paths at a time t_k
$\mathbb{APIS}_{t_k}$	is a finite set of all attack paths with IP-shuffling at a time t_k
$\mathbb{SAP}_{t_k}$	is a finite set of all the shortest-attack paths at a time t_k
$\mathbb{SAPIS}_{t_k}$	is a finite set of all the shortest-paths with IP-shuffling at a time t_k
NAP_{t_k}	is the number of attack paths at a time t_k
NAV_{t_k}	is a network address variability at a time t_k
NAV	is a network address variability on scanning time window $[t_1, t_m]$
$NAV_{t_k}^U$	is a network address variability with a union of all previous states at a time t_k
HAV_{t_k}	is a host address variability at a time t_k

HAV	is a host address variability on scanning time window $[t_1, t_m]$
$HAV^U_{t_k}$	is a host address variability with a union of all previous states at a time t_k
APV_{t_k}	is an attack path variability at a time t_k
APV	is an attack path variability on scanning time window $[t_1, t_m]$
$SAPV_{t_k}$	is the shortest attack path variability at a time t_k
$SAPV$	is the shortest attack path variability on scanning time window $[t_1, t_m]$
$SAPVIS_{t_k}$	is the shortest attack path variability with IP-shuffling at a time t_k
$SAPVIS$	is the shortest attack path variability with IP-shuffling on scanning time window $[t_1, t_m]$
$VNAP_{t_k}$	is a variability of the number of attack paths at a time t_k
$VNAP$	is a variability of the number of attack paths on scanning time window $[t_1, t_m]$
SSP_{h_i}	is a scan success probability of a host h_i
ESP_{h_i}	is an exploit success probability of a host h_i
CSP_{h_i}	is a compromise success probability to attack a host h_i
CSP_{t_k}	is a compromise success probability to attack a host h_i on AP_{t_k} at a time t_k
CSP	is a compromise success probability to attack a host h_i on scanning time window $[t_1, t_m]$

Chapter 1

Introduction

1.1 Background

Conventional networked systems have been characterized by static system configurations, which can significantly provide benefits to attackers in terms of their resource utilization in time and effort. The attackers often enjoy asymmetric advantages because they can take enough time to investigate a target system by collecting their configuration information to identify exploitable vulnerabilities. Based on the obtained intelligence towards the system configuration, the attackers can plan to launch their attacks in order to maximize their utility and success.

Moving Target Defense (MTD) is a security defense mechanism that dynamically changes attack surfaces of a target system in order to increase uncertainty and confuse the attackers, resulting in invalidating the attacker's intelligence or information collected during its attack reconnaissance (e.g., hosts and services scanning attacks) or procedures to launch targeted attacks. Unlike the concept of the conventional security mechanisms aiming to eliminate any security vulnerabilities from a system, MTD can work with legacy resources by changing the system configurations (e.g., IP/port addresses, software stacks, OS, virtual machines, network topology) that can be used as an effective cybersecurity defense mechanisms [62, 74, 81].

Randomization of network and/or system's configurations (e.g., IP, ports, route *etc.*) has been researched as MTD approaches to thwart reconnaissance and scanning attacks [74, 99]. This randomization approach has been often applied at different levels of the network and/or system to change their properties [119]. For instance, instruction set randomization [83], address space layout randomization [137], IP address randomization or random host mutation [3], and random route mutation [44] are common MTD approaches based on the randomization. Therefore, randomness is a key characteristic of MTD mechanisms to ensure their effectiveness as it provides unpredictability, which is an important goal of the MTD mechanisms. A high degree of unpredictability can make harder for attacker to predict the next state of the network and/or system's configurations, which increases attack effort and reduces the attack success. Most of MTD mechanisms have this randomness feature, where the defender chooses a next sequence of the configurations pseudorandomly to make them unpredictable [22]. The degree of randomness or unpredictability can be increased by using advanced and cryptographically secure algorithms.

The conventional networked systems are composed of heterogeneous elements such as switches, firewalls, and routers with their proprietary software and protocols. This kind of network setup has very little flexibility and scope for innovations and researches in the networks. *Software-Defined Networking* (SDN) has emerged as a solution for resolving this problem of these existing networks [136]. The SDN architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and underline infrastructure to be abstracted for applications and network services [88]. These flexibility and programmability features of the SDN technology can be applied to several cybersecurity applications, including MTD techniques. For instance, the SDN controller (e.g., Ryu [133], OpenDaylight [108]) can enable a switch to rewrite the packet headers and steer the traffic towards a specific host by merely updating the forwarding table of

the switch via an OpenFlow protocol [107]. This flexibility of the network programming (e.g., packet header modification, dynamic address management, rerouting) provided by the SDN technology can facilitate the implementation of the MTD operations for cybersecurity. Also, leveraging the SDN technology can help to reduce both capital expenditures (CAPEX), and operational expenditures (OPEX) costs [82]. Flexible placement of the network functions, fine-grain network traffic optimization, orchestration of network resources, and automation of the ongoing operational and maintenance tasks all provide an intuitive indication of potential reduction of the costs [61, 82].

MTD research in SDN-based environments is in an infant stage since both MTD and SDN research areas have been emerging relatively recently due to their respective technological advantages (e.g., proactive, dynamic, and adaptable nature of MTD, and flexibility and programmability features of SDN). Enhancement of current SDN-based MTD research and/or development of novel MTD mechanism would be surely a valuable addition to the current security researches; and development of novel security metrics, particularly capturing dynamic networks and/or systems and attack behaviors, is a novel research direction in which most existing security metrics mainly focus on measuring static aspects of networks and/or systems.

1.2 Problem Statement

The homogeneous and statically configured networked systems provide asymmetric advantages to attackers, which make them easier for performing reconnaissance and/or launching their attacks. Also, the attackers have enough time to investigate the target systems and plan the attacks. The security attacks cause tremendous socio-economic impact and losses, therefore, it is a crucial need to not only respond to the attacks but prevent them in advance. MTD mechanism creates an asymmetric uncertainty to the attackers by continuously changing the attack surface, which aims to increases uncertainty

and complexity, to decrease opportunities for identifying the targets, and to introduce higher cost in launching the attacks [64, 114]. The MTD mechanism acts as an intrusion prevention mechanism, which reduces the risk of potential intrusions to the system by changing the attack surface and minimizing the potential of the attackers in discovering the vulnerable network or system components, and their attempts to penetrate into the system. Unlike, conventional security mechanism that aims to keep the system in a perfect secure by detecting and eliminating any security vulnerabilities from the system, MTD mechanism disrupts attacker by continuously and dynamically changing the network or system configurations [77, 74].

The dynamic defense mechanisms based on MTD involve randomizing network or system components (e.g., IP address, TCP/UDP ports, network topology, VM, software) to reduce the likelihood of successful attacks. The MTDs can be categorized into the five different domains (e.g., data, software, run-time environment, platform, and network) according to their place within the execution stack [118]. Network address shuffling is one of network-based MTD mechanisms that dynamically change an IP address or a port number of a host or service of a target system [21]. IP address shuffling (e.g., [3, 73, 10, 74, 75, 157, 99]) and port hopping (e.g., [93, 143, 100]) are widely used MTD mechanisms based on IP addresses and/or ports shuffling. These defense mechanisms change the IP or virtual IP address or port numbers of the target system at a certain fixed time interval (e.g., in every 30s). However, there are problems in network address shuffling MTD mechanisms. First, these MTD mechanisms randomly mutate or shuffle IP addresses of hosts, mapping them to virtual IP addresses in a one-to-one manner. One-to-one mapping requires more IP addresses to satisfy the mutation rate constraint and unpredictability for the hosts in the network [74], which often results in a lack of scalability due to a limited address space. Therefore, it needs to develop new IP address shuffling MTD approaches for flexible mapping and shuffling the virtual IP addresses of the server machines of the SDN in effectively and efficiently.

Secondly, in a conventional network, a server machine of the network commonly provides services using well-known TCP/UDP ports, which remain unchanged during the service life. The services are vulnerable and long-time exposure to potential attackers. There are many IP and/or port address shuffling-based MTD approaches (e.g., [157, 99, 93, 143, 100, 3, 73, 10, 74, 75]) have been proposed to address aforementioned problem. However, the proposed approaches lack to use of both IP addresses and ports into a single security defense mechanism. With the shuffling of both IP address and ports able to hide real services and host's identities, which can create an extremely diverse environment and makes more difficult to attackers for discovering vulnerable hosts and services. Also, the address mapping of the approaches is 1-to-1 manner, and they often deployed in conventional networks. Hence, it is important to propose new proactive MTD mechanisms for shuffling both IP addresses and ports of the hosts in the SDNs.

Thirdly, many security assessment models and metrics have been proposed for measuring the effectiveness of the MTD techniques [166, 166, 150, 164]. Hong and Kim [64] proposed metrics, such as system risk, attack cost, and reliability for assessing the effectiveness of MTD categorizing into three different classes, shuffling, diversity, and redundancy, respectively. Similarly, Hong et al. [66] proposed attack and defense effort-based metrics for assessing the effectiveness of the MTD techniques. However, the proposed metrics do not offer the ability to capture and measure the effectiveness of the network address shuffling-based MTD techniques. Besides, the existing metrics are very application-specific, limited to particular attack scenarios and models, and lack to capture and measure the effectiveness of the various types of MTDs. Therefore, it is necessary to design a new set of dynamic security metrics that can capture network dynamics and measure the effectiveness of deploying MTD mechanisms.

1.3 Research Questions and Goals

This thesis takes the development of MTD mechanisms and dynamic security metrics approach to answer the following research questions:

- Q1: How can we shuffle virtual IP addresses of server hosts in the SDN?
- Q2: How can we protect the SDN with the existence of the vulnerable services on the server systems in the SDN?
- Q3: How can we measure the effectiveness of moving target defenses in the SDN?

The goals of this thesis are to advance the development and evaluation of moving target defenses for the software-defined networks. Three goals corresponding to the research questions are described as follows:

- G1: *To develop a new moving target defense mechanism that continuously and dynamically changes the virtual IP addresses of a server host in the SDNs.* The outcome of this goal includes a new proactive SDN-based MTD mechanism that enables the server host to have multiple, random, time-varying virtual IP addresses which are changed frequently with a specified time interval in order to increase attacker's work effort invalidating target system's information collected by the attacker through network scanning.
- G2: *To design and develop a moving target defense mechanism that protects the SDN by shuffling both hosts' virtual IP addresses and services' port numbers in the SDN.* The outcome of this goal consists of a new dynamic and proactive SDN-based MTD approach using the shuffling of virtual IP addresses and port numbers aiming to obfuscate both network and transport layers' real identities of target server host and the vulnerable services for defending against the network reconnaissance and scanning attacks.

- G3: *To develop a set of dynamic security metrics for measuring the effectiveness of the SDN-based MTD techniques.* The outcomes of this goal include a new suite of dynamic security metrics to capture and measure the dynamics of a network and/or a host security information (e.g., IP addresses, port numbers, software stacks, vulnerabilities, network topology *etc.*) introduced by various types of MTD techniques deploying in the SDNs.

1.4 Methodology

We define a methodology with five phases to answer the proposed research questions systematically. Five phases of the methodology are described below:

System Model: An SDN-based enterprise network is used as the system model that comprises an SDN-controller (e.g., Ryu [133]), SDN-switches (e.g., Open vSwitch [125]), end-hosts (e.g., server-hosts, client-hosts), firewalls/routers *etc.*. The SDN controller implements the proposed defense mechanism (e.g., MTD) and controls data forwarding and routing operations of the network elements using an Openflow-protocol [107] in a centralized manner. It is assumed that networked systems' configurations (e.g., IP addresses, ports, vulnerabilities, topology, route) are changed dynamically by the deployed defense mechanism in every interval time (e.g., MTD interval time). The changes in the configuration's information are captured at different points of time and used as input to the metrics calculations.

Attacker Model: We define the attacker model to describe how the attacker interacts with the system model and what types of the attacker and attacks are considered. We make the assumptions for the attacker's goals and capabilities, knowledge of the target systems and attack tools used, attack strategies, and potential attacks. In this thesis, we assume that the attacker is located outside the system and attempts to enter into the system with entry points, targets, and goals stages. The potential attack scenarios are

considered and modeled with various attack stages (e.g., discover, exploit, and compromise) for the different attack strategies of the attacker.

Defense Model: We develop the defense model, which consists of the proactive and dynamic defense approaches to protect the security of the network based on the SDN. These defense mechanisms are designed and developed to solve the research questions which can be used to change attack surface dynamically over time. They validated by comparing them with a baseline model based on the predefined system and attacker models. The baseline model is a typical static network where the configuration of the network remains unchanged.

Metrics & Measurements: A suite of dynamic security metrics are developed to measure the effectiveness of the MTD mechanisms deployed in the SDN-based network environments. These metrics capture the dynamics of the network system made by the deploying MTD mechanism using graphical security models at different points of time. The metrics use system model, attacker model, and defense model as input. The metrics are computed based on the network reachability and host's vulnerability information of the system model over time. The attacker and defense models specify the capabilities and goals of the attacker and the deployed defense mechanisms (e.g., MTD) in the system model, respectively.

Evaluations: We evaluate the proposed approaches via probabilistic analysis and simulations methods to validate their feasibility and performance. In a probabilistic analysis model, the behaviors of the system model, the attacker model, and the defense model and the interactions between them are described based on probabilistic parameters and evaluated. The evaluation of probabilistic models provides insights and lessons based on the observations of general system behaviors with minimum evaluation cost. The simulation methods evaluate and validate the performance of the proposed approaches based on simulation experiments. The simulation scenarios and experiments are considered based on the system model, the attacker model, and the defense

model. Emulation models can provide a higher validity since it provides flexibility in modeling realistic attack types, networked systems, and MTD techniques.

1.5 Research Contributions

Four main contributions are proposed to moving target defenses in software-defined networks, which are as follows.

1. **Development of Flexible Random Virtual IP Multiplexing in SDNs**: We propose a flexible, dynamic, proactive software-defined networking (SDN)-based MTD technique called *Flexible Random Virtual IP Multiplexing*, namely FRVM, which aims to thwart network reconnaissance and scanning attacks (i.e., it has been published in [139]). FRVM enables a host machine to have multiple, random, time-varying, virtual IP addresses, which are multiplexed to a real IP address of the host. FRVM frequently changes all the virtual IP addresses of a specified time interval; the multiplexing and de-multiplexing operations of it remaps the virtual IP to a real IP, and a real IP to the virtual IP address(s) of the host respectively. Therefore, at the end of every multiplexing event, the FRVM aims to make the attackers lose any knowledge gained through the reconnaissance and to disturb their scanning strategy. Since identifying IP addresses in a target domain is a precursory step for most attacks, FRVM can be used to prevent the attackers from scanning strategy.

2. **Security performance and overhead evaluation of the MTD in SDNs**: We comparatively analyze and evaluate the performance of the SDN-based MTD (e.g., FRVM) with a baseline model (i.e., a typical static network without FRVM) through the virtualized testbed using Mininet [91] and simulation experiments using the security and

performance metrics. We investigated the performance of the FRVM in terms of attacker success probability under scanning attacks and defense costs, considering the end-to-end delay and throughput overheads. Also, we analyze the flow management requirements of the FRVM to deploy in the SDN-based network environments.

3. **Development of Random Host and Service Multiplexing in SDNs**: We propose *Random Host and Service Multiplexing* (RHSM) as a novel MTD technique to be deployed in an SDN environment to defend against network reconnaissance and scanning attacks with the aim of effectively minimizing attack success probability (ASP) under the various levels of the scanning attacks and server-host vulnerabilities which has been published in [140]. Also, we investigate overhead derived from operating RHSM in terms of the size of the flow table, the frequency of flow rules modifications, and the performance of an SDN controller. We develop an SDN-based architecture, algorithms, and communication protocols that enable the more effective realization of the proposed RHSM on the testbed. To validate the performance of RHSM, we compare RHSM with a baseline model with a static network configuration in terms of the ASP. We analyzed the effect of port address space scanned by attackers under varying the number of service vulnerabilities. Our results proved that the proposed RHSM could effectively thwart scanning attacks and provide a proactive and resilient security solution by effectively hiding the real identities of both a server-host and its active services in an SDN environment.

4. **Development of Dynamic Security Metrics for measuring the effectiveness of the MTDs in SDNs**: We propose a suite of dynamic security metrics that timely, dynamically, and adaptively assess the effectiveness of the software-defined network (SDN)-based moving target defense (MTD) techniques. The security metrics are developed to

measure the dynamics of a network and a host state's information (e.g., IP address, port, software stacks, vulnerabilities, or network topology) introduced by various types of MTD techniques shuffling them. The key aspect of our proposed metrics is to capture variability that keeps track of changing patterns of the network and the host states upon every MTD triggering event. In this work, we propose the following security metrics capturing the variability based on the changes made by the MTD:

- *Network and host address-based metrics* measure the variability of the network and the host addresses based on a degree of uncertainty and unpredictability on the assigned IP address to the hosts in a network;
- *Attack path-based metrics* compute variability of attack paths using graphical models estimated based on the network state transitions from one topology to another topology upon triggering a network topology or IP shuffling MTD; and
- *Attack stage-based success metrics* measure the chances of discovering a vulnerable target host's information, exploiting the target host's vulnerability, and compromising the target host.

Besides, we investigated the key parameters that can significantly affect the MTD performance based on the proposed dynamic security metrics.

1.6 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 discuses state-of-the-art MTD researches, including key concepts, classifications, attacks modeling, and metrics used in MTD. Chapter 3 presents the description of the proposed Flexible Random Virtual IP Multiplexing (FRVM) defense mechanism and evaluation of the FRVM via simulation experiments in a virtualized network environment using *Mininet* [91] emulator (addressing

the research goal G1). Chapter 4 proposes the Random Host and Service Multiplexing (RHSM) mechanism for MTD in SDNs (addressing the research goal G2). Chapter 5 describes the dynamic security metrics for measuring the effectiveness of the MTD techniques in SDNs (addressing the research goal G3). Chapter 6 discusses the usability and limitations of the thesis and highlights the possible future research directions. Finally, Chapter 7 concludes the thesis.

Chapter 2

Literature Review

This chapter discusses the related work on state-of-the-art MTDs, including MTD concepts, classifications of MTD, attacks modeling in MTD, and metrics used for assessing the effectiveness and efficiency of the MTD mechanisms. Section 2.1 presents the MTD concepts, including the evolution of MTD, key roles and design principles of MTD, defensive deceptions, and the existing SDN-based MTD approaches. Section 2.2 describes the classifications of the MTD techniques, Section 2.3 discusses the attacks considered in the MTD techniques and the existing MTD metrics are presented in Section 2.4.

2.1 MTD Concepts

In this section, we provide backgrounds and key design principles of MTD. We discuss the two major roles of the MTD in terms of the intrusion prevention mechanism and the detection of potential attackers. Also, we clarified the design principles of the MTD with three key questions and summarized the benefits and caveats of applying the MTD techniques.

2.1.1 Evolution of MTD

In the classic "shell game," dating at least to the times of ancient Greece, a pea or ball is hidden under one of three shells or cups, and players gamble to guess the location of the pea after the shells have been moved. This game is also known as "thimblerig," "three shells and a pea," and the "old army game" and has variant forms, such as "Three-card Monte." The advantage for the operator of the game is that sufficient movement of the shells will confuse the players of the location of the pea. This underlying idea exemplifies the philosophy of the so-called "Moving Target Defense (MTD)" strategy in cybersecurity [114].

The basic axiom of MTD is that it is impossible to provide complete and perfect security for a given system. Given this, the objective is to enable the normal functioning of the system (i.e., the normal provision of services) even in the presence of malicious actors seeking to compromise the system. Since attacks cannot be prevented, the goal is to defend against and thwart attacks. In practice, this goal is achieved via MTD by the manipulation of multiple system configurations to modify and control the attack surface, where the attacker engages with the system. MTD aims to increase uncertainty and complexity for any attacker of the system, to decrease the opportunities for the attacker to identify targets (e.g., vulnerable system components), and to introduce higher cost in launching attacks or scans (e.g., reconnaissance attacks). The desired result is that the attacker will waste time and effort without gaining useful intelligence about system [64, 114].

Origins of the conception of MTD can be found under different names in the computer security literature. Some of these research areas include fault tolerance (or reliability using redundancy) even since the 1970's (e.g., n-version programming (NVP) [12, 27]), reconfigurable computing (e.g., reconfigurable software [34], network reconfiguration [15]), and/or bio-inspired cybersecurity (e.g., software and/or network diversity for cybersecurity [87, 90]). In 2009, the Networking and Information Technology Research and Development (NITRD)

Program explicitly emphasized the concept of MTD in terms of its effectiveness and efficiency, which can leverage existing resources [57].

2.1.2 Key Roles of MTD

MTD can contribute to enhancing the security provided by an intrusion defense system (IDS) in two ways. First, MTD acts as an intrusion prevention mechanism by reducing the risk of potential intrusions to the system. Many MTD techniques, including Internet Protocol (IP) and/or TCP ports randomization and/or mutation, network re-configurations, topology and/or route changing, platform diversity (e.g., OS, VM migration *etc.*), and software stack shuffling *etc.* changes the attack surface, thereby minimizing the potential of attackers in their discovery of vulnerable system components and in their attempts to penetrate into the target system.

Second, MTD can assist in the detection of potential attackers. Since the MTD techniques can increase the complexity of an attack operation to identify a target, the intelligence about attack patterns or behaviors of potential attackers can be easily obtained by monitoring active activities (e.g., scanning) by the potential attackers. This allows MTD techniques, categorized as intrusion protection mechanisms, to also indirectly assist an intrusion detection system in improving the detection of potential attacks. As Table 2.1 describes, compared to the characteristics of the conventional security techniques, MTD aims to provide *affordable, service-oriented defense* in terms of minimizing defense cost (e.g., communication or computation overhead), maximizing service availability to users, and meeting a required level of security. In particular, the underlying idea of MTD lies in 'affordable defense' because it focuses on rearranging system configurations to confuse attackers while still employing existing security mechanisms. This means that MTD does not require the development of new security mechanisms, which often need to goadopt through tough, high-standard security analysis and accordingly may require an excessive amount of time or effort. Therefore, MTD provides a

Table 2.1: Comparison between the conventional defense and MTD mechanisms.

	Conventional defense mechanism	**MTD mechanism**
Philosophy	Building a secure system by covering all vulnerabilities is the key for security	It is impossible to build a perfectly secure system
Goal	Make a system perfectly secure	Thwart attackers by changing attack surface
Defense Type	Static defense using static system configurations fortifying assets and hardening defense systems	Dynamic defense using dynamic system configurations to build an agile system
Key Concern	How to identify all potential system security vulnerabilities and eliminate them	How to optimally execute MTD operations to continuously change attack surface while meeting multiple system objectives concerning security and performance
Challenges	Due to limited capability to detect all exhaustive, possible vulnerabilities and their high complexity, and it is unfeasible to identify all possible vulnerabilities	Optimally executing MTD operations is not a trivial task because frequently executing MTD operations are costly and often hurts service availability to users

new way of defense that continues to leverage legacy resources but enhances security by dynamically changing the attack surface, such as any aspects of system components (e.g., data, software, platforms, run-time environments, networks) to increase uncertainty and confusion for the attackers.

2.1.3 Key Design Principles of MTD

MTD was introduced as a proactive defense mechanism to prevent cyberattacks continuously and dynamically changing the attack surface of systems [57]. The fundamental design principle for developing the MTD techniques lies in the decisions for the following three key questions [22]: *what to move*, *how to move*, and *when to move*.

What to move: 'How to move' refers to what system configuration attribute (i.e., attack surface) can be dynamically changed to confuse attackers. The example system or network attributes that can be changed include

instruction sets [83, 127], address space layouts [137], IP addresses [3, 10, 73, 85], port numbers [99], proxies [79], virtual machines [170, 18], operating systems [151], or software programs [71]. A change in the system configuration attributes must cause a change on the attack surface of the system leading to the loss of work and increased complexity for attackers. Additionally, the number of possible values the dynamic attribute can change to must be vast, thwarting the attacker's ability to simply brute-force the future value [22].

How to move: 'How to move' defines how to change the moving attributes (i.e., targets) to increase unpredictability or uncertainty, leading to an attacker's high confusion. This is related to the three operation-based MTD classification components: shuffling, diversity, and redundancy (SDR). Common MTD techniques include artificial diversity [18, 68, 71, 132, 154] or randomization [3, 10, 73, 79, 83, 85, 99, 103, 137, 144, 158], which rearrange or randomize the various system and network attributes where each work can belong to shuffling, diversity, and/or redundancy techniques. Fig. 2.1 depicts examples of how to move the moving elements using MTD based on the *shuffling, diversity and redundancy* classification [4, 64].

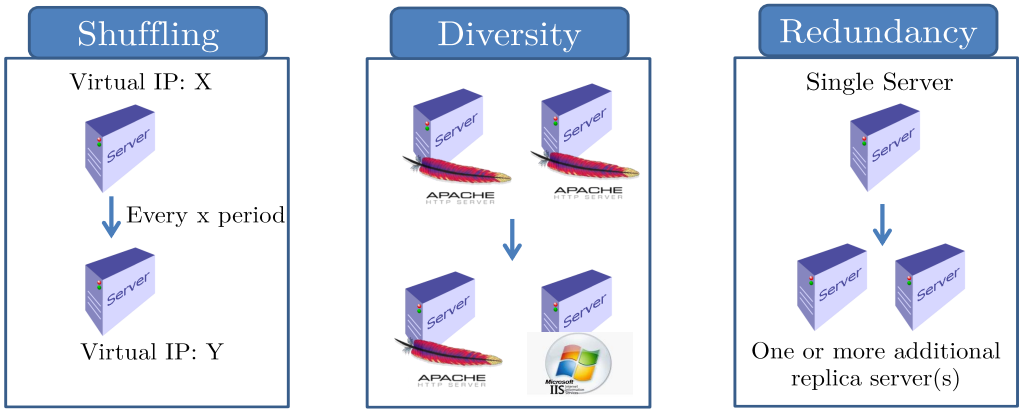


Figure 2.1: Examples of the three MTD techniques: shuffling, diversity, and redundancy.

When to move: 'When to move' involves deciding the optimal time to change from the current state of an MTD system to a new state, invalidating information or progress gained by an attacker in the current state. The three

common types of movement (or adaptation) are reactive, proactive, or hybrid strategies.

- **Reactive movement**: The reactive movement is based on an event or an alert that executes the adaptation as a response to a message from the detection of suspicious activity. Reactive MTD attempts to counter actions taken by an attacker under the assumption that the attacker's actions are detected.

- **Proactive movement**: The proactive movement triggers MTD techniques by changing configurations or properties on a schedule, with either fixed or random intervals between adaptations. The proactive movement ensures that any information gained by the attacker quickly expires [103], thus regular movement is important. Proactive MTD moves regardless of the presence of an attacker costing additional delay in the protected system.

- **Hybrid movement**: The hybrid movement is based on both reactive and proactive features, wherein the time interval to trigger an MTD operation is adaptive upon certain events or security alerts while the interval is also bounded in length to prevent potential, undetected security threats [175].

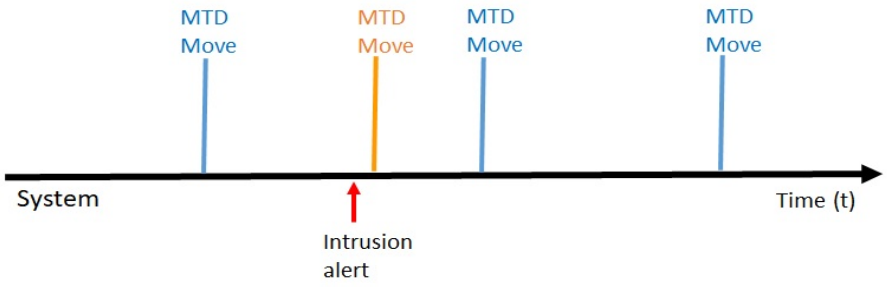


Figure 2.2: An example of the MTD move with proactive and reactive MTD movement.

Fig. 2.2 shows an example of 'when to move' which addresses at what time point the element(s) of MTD can be moved under hybrid movement [4, 64]. In

Fig. 2.2, a blue line means an MTD can be triggered at a fixed MTD interval, whereas an orange line shows an MTD triggered by an event (e.g., an intrusion detection alert). Intuitively, we can notice that using the fixed MTD interval is more proactive than using the adaptive MTD interval in confusing attackers. However, the high frequency of MTD operations may incur high costs. Hence, an optimal MTD interval can be identified at run-time to balance both cost and security. A running system can have different moving elements (e.g., IP, Port, OS, VM, Applications) over the time, and their changes can take place either at a regular fixed interval of time or upon receiving an intrusion detection alert.

2.1.4 Defensive Deceptions

In military settings, deception is originally derived from actions to be executed to intentionally mislead an opponent's decision associated with the strengths and weakness of military capacities, intents, or operations. Defensive deception is created to affect an attacker's action in a way a defender wants to achieve its mission [141]. Deception has been studied based on many classifications. Bell and Whaley [17] and Almeshekah and Spafford [8] discuss deception techniques in terms of either 'hiding the real' or 'showing the false.' That is, the defender can modify the information or present misleading information to the attacker. For example, the defender can mislead the attackers to misjudge by providing a false sense that the attacker has complete, certain, relevant information for its decision making, which is not true indeed. Caddell [20] classifies deception in terms of 'passive deception' and 'active deception.' The passive deception includes actions to hide capabilities or intents from the attacker. The active deception involves leading the attacker to form false beliefs by providing false information. Daniel and Herbig [42] introduces two types of deception techniques based on their goals, aiming to increase an attacker's perceived ambiguity and to mislead the attacker to misjudge. Rahman et al. [129] proposed a game-theoretic approach using the Nash equilibrium strategy as a deception technique to prevent remote

OS fingerprinting attacks. They showed that the designed technique named *DeceiveGame* could significantly decrease the fingerprinting attack success probability while the overall usability of the system is preserved without performance degradation. Bell and Whaley [17] and Almeshekah and Spafford [8] show some other example deception techniques that dissimulate information in order to hide real information and simulate information to make the attacker form false beliefs. Examples of dissimulation deception are masking (i.e., hiding the real), repackaging (i.e., creating the false), and dazzling (i.e., confusing an attacker) [8, 17].

MTD and deception, both defense approaches have the same goal to confuse attackers by increasing their uncertainty in the decision process of launching attacks. Hence, the deception techniques used for increasing confusion can also belong to the MTD. Also, even if the deception techniques are used to mislead attackers, but they are also being dynamically deployed (i.e., a different set of decoy nodes is deployed at a different time [54]), they can be treated as MTD techniques.

Although these two defense techniques are common in their goal, deception takes a more aggressive strategy than MTD in terms of intentionally presenting false information for the attacker to form false beliefs while MTD does not use a strategy of lying or disseminating false information to mislead attackers. However, these two defense mechanisms can work together well to benefit each other in the following sense. Deploying deception can be less costly than MTD operations and can help MTD to save its cost because the attack can be delayed if an attacker is successfully deceived by the deployed defensive deception and accordingly, MTD does not have to be triggered as often as the system without deception [29]. Besides, the defense strength achievable by defensive deception is limited in nature because the deception can be ultimately known to the attacker sometime later. When the attacker found it was deceived by the defensive deception, the system can trigger an MTD operation to protect the system without a security breach.

2.1.5 SDN-based MTD Approaches

Software-defined networking has emerged as a promising technology to decouple the network control plane from the data-forwarding plane for providing flexibility, robustness, and programmability to a networked system [136]. In conventional networks, a routing algorithm in each switch makes packet forwarding decisions. By contrast, a controller on an SDN is designed to control the forwarding operations of the switches in a centralized manner. Various cybersecurity network applications have leveraged the flexibility and programmability of SDN technology. In this section, we discuss the existing MTD techniques, attacks, methodologies, and evaluation methods used to develop MTD technologies to protect SDN-based network environments.

IP shuffling / mutation-based MTD is popularly used in SDN environments by leveraging OF switches and a centralized SDN controller [74, 73, 103] where the SDN controller makes packet-forwarding decisions at each OF switch and decisions on the updates of the flow tables at the switch. Network topology shuffling is also a well-known MTD technique [1, 13] that has been applied in SDN environments in order to minimize security vulnerabilities in attack paths [64] by the SDN controller identifying an optimal network reconfiguration. The scalability issue in SDN environments is also managed by the SDN controller, which creates scalable attack graphs [30]. An SDN-based packet header randomization is proposed to realize unlinkability anonymity in communications where the SDN controller and OF switches take care of routing based on nonce information [144, 158]. The common attacks considered in SDN-based MTD approaches include reconnaissance (or scanning) attacks [30, 63, 64, 74, 73, 81] and DDoS attacks [13, 147], which can be countermeasured by using random IP mutation and/or network topology shuffling.

The key idea of deploying MTD techniques in SDN environments is to

highly leverage its centralized structure with an SDN controller to optimize the configuration of the deployed MTD techniques, such as IP randomization / shuffling [74, 73, 103, 147], network routing paths [65], attack graphs / paths [30], port hopping [81], packet header randomization / obfuscation [144, 158], or virtual topology generation [2]. The key concerns in developing SDN-based MTD approaches are resolving a scalability issue in an attack graph [30] or IP shuffling [103] and optimizing both security and performance in terms of minimizing security vulnerabilities while minimizing defense cost and service interruptions to users. Simulation-based validation methods are popular, but some emulation-based testbeds are also developed for validating SDN-based MTD (e.g., a *Mininet* emulator with NOX SDN controller [73]). Hong et al. [65] conducted security and performance analysis of their proposed network topology shuffling-based MTD in a real SDN testbed consisting of SDN-enabled hardware switches and an SDN controller. Kampanakis et al. [81] showed the performance of the proposed SDN-based port hopping MTD based on Cisco's One Platform Kit.

2.1.6 Discussions on Benefits and Caveats of MTD

In this section, we discuss the key benefits of MTD and caveats in developing MTD techniques concerning the roles and design principles of MTD techniques.

The **key benefits** of MTD techniques include as follows:

- *Providing affordable defense opportunities.* Compared to conventional security mechanisms aiming to perfectly eliminate any vulnerabilities and risks that can be introduced by attackers, MTD provides a new perspective of a defense system by continuously changing the attack surface, which makes it harder for attackers to achieve their goals. MTD allows us to leverage legacy system components and existing technologies, enabling a greater likelihood of achieving affordable defense solutions and avoiding the necessity of creating a new, highly robust security

mechanism, such as cryptographic solutions, which requires more resource / time to develop but may be less applicable in broad contexts (e.g., under the assumption of trusted entities for key management).

- *Providing another layer of defense in cooperation with existing defense mechanisms.* At the same time, MTD can cooperate with other defense mechanisms by assisting in intrusion detection that can thwart potential attackers and/or provide new attack patterns observed during the reconnaissance stage of the attack. Also, MTD can add another layer of defense when a defensive deception is used to deceive an attacker. In particular, when the attacker realized the deception and can successfully identify vulnerable system component, not being lured by the deception (e.g., a honeypot), MTD can be executed to migrate the system platform in order to invalidate the information of the system vulnerabilities collected by the attacker in the current system configuration.

However, we also need to be aware of the following **caveats in developing MTD techniques**:

- *High frequency of triggering MTD operation may significantly hinder seamless, quality service provision to users.* This issue is related to a design principle related to 'when to move' a target because triggering MTD operations too often naturally leads to reducing service availability and/or interrupting seamless service provision.

- *Non-adaptive, non-intelligent MTD may waste defense cost while not decreasing system vulnerabilities.* In some system/network settings, resources are highly constrained, such as IoT environments or mobile, wireless networks. The strategy in deciding how to expend defense resources is critical to prolonging system lifetime and increase system reliability (or survivability). If a deployed MTD is not adaptively and/or intelligently executed based on a level of detected security vulnerabilities,

then the required system goals, such as minimizing defense cost and maintaining a certain level of system security, may not be met.

- *How to execute MTD operations in resource-constrained, distributed network environments has not been sufficiently studied in the literature.* Most existing MTD approaches consider a trusted third party for network management associated with running an MTD technique. However, in some decentralized or distributed, resource-constrained environments, such as mobile ad hoc networks, wireless sensor networks, or IoT networks, it is not easy to identify a trusted infrastructure that can make critical decisions on MTD executions. Further, the cost of executing MTD operations should be considered as a priority as those contested environments can only afford highly lightweight MTD solutions.

2.2 Classifications of MTD

MTD techniques have been studied under various classifications with different criteria. In this section, we discuss how the MTD techniques have been classified in the literature. Also, we distinguish the concepts of MTD from those of deception and discuss what the commonalities and differences are between them.

2.2.1 Timeliness-based MTD

Timeliness-based MTD classifies MTD techniques based on criteria to determine the MTD strategy 'when to move.' Fig. 2.3 depicts the three common timeliness-based MTD categories as follows:

2.2.1.1 Time-based MTD

This approach triggers an MTD operation based on a certain time interval called the *MTD interval*, which can be a fixed interval or a variable interval [22].

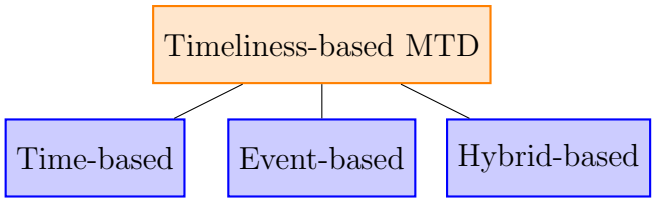


Figure 2.3: Classifications of timeliness-based MTD techniques.

With a fixed time interval, the MTD mechanism periodically changes the attack surface (e.g., IP/Port addresses, OS rotation, VM migration) with a constant equal time, which remains unchanged. If the interval time is too large, then an attacker may be allowed a sufficient amount of time to scan a system and then penetrate the system, resulting in a security breach. On the other hand, if the interval time is too short, then the MTD triggers even when there is no attack on the system, wasting defense resources and degrading performance. Therefore, determining the MTD interval time to perform an MTD operation has a significant impact on the effectiveness and efficiency of a given MTD technique [23].

2.2.1.2 Event-based MTD

This approach performs an MTD operation only when a certain event occurs. The event can encompass any indication an attacker accesses a system or attempts to launch a certain attack. That is, if the defender can accurately predict potential attacks, then events can trigger the appropriate MTD operation. In order to discover the key events that should trigger the MTD action, attack prediction based adaptive MTD approaches have been proposed using machine learning [33], game theory [172], and control theory [132].

2.2.1.3 Hybrid MTD

Some MTD approaches take a hybrid strategy. These approaches [84, 132, 174, 175] execute MTD operations adaptively based on both the time and event-

based MTD strategies for proactive and reactive adaptations, respectively.

2.2.2 Operation-based MTD

In this section, we study MTD techniques and classify them based on criteria to determine 'how to move' as described in Section 2.1.3. Hong and Kim [64] classified MTD techniques into three classes based on the nature of their operations: *shuffling*, *diversity*, and *redundancy*.

2.2.2.1 Shuffling

This technique rearranges or randomizes system configurations, such as mutating IP addresses at a TCP/IP layer or dynamically adjusting the migration time of VMs. The key goal of these shuffling-based MTD techniques is to increase confusion and uncertainty for attackers (i.e., to make the identification of vulnerable targets more difficult) by making information collected by the attackers obsolete or by wasting attackers resources in the collection of useless information. Ultimately, shuffling-based MTD can prevent or delay the attackers from accessing a target system. Because the system earns more time to monitor the attack behaviors (e.g., scanning attacks), the system's defense mechanisms (e.g., IDS) can prepare more intelligent strategies to deal with the attack based on identified attacks. In this section, we discuss various methods which have been used to change the system and/or network configurations or attack surface.

- **IP-Shuffling / Mutations**: Many shuffling-based MTD techniques used IP shuffling or mutation in various network domains. Kewley et al. [85] proposed an IP hopping method to defend against any adversaries sniffing the network by translating an IP address before any packet enters to a public network. It hides the IP address to prevent man-in-the-middle sniffing attacks. However, it did not concern scanning attacks relying on probe responses to discover client hosts. Antonatos et al.

[10] proposed the NASR scheme to deal with hitlist worms. NASR is a LAN-level network address randomization technique based on the dynamic host configuration protocol (DHCP) updates. Although NASR reflects the nature of shuffling based on the IP address randomization as an MTD mechanism, it disrupts the active connections and has limitations in providing high unpredictability and mutation speed due to constrained address space in terms of a LAN address. Shi et al. [143] developed an active cyber-defense mechanism based on port and address hopping with timestamp-based synchronization. However, this cyber-defense mechanism is developed to deal with denial-of-service (DoS) attacks only within the scope of metropolitan area networks (MANs) and multiple LANs. Jafarian et al. [73] proposed an IP address shuffling technique called open-flow random host mutation (OF-RHM) that randomly mutates the IP address of the end-hosts. In OF-RHM, a real IP address of the end-host remains unchanged while associating with a short-lived virtual IP address, which has been changed randomly and periodically. Although the OF-RHM is transparent to end-hosts and provides a high mutation rate, it cannot be deployed in a conventional network due to low scalability. Al-Shaer et al. [3] proposed random host mutation (RHM) to address the scalability problem in conventional networks. RHM uses a two-phase mutation approach with low and high-frequency mutation intervals for allocating virtual address range and selecting *vIP* to the host, respectively. However, this work did not consider the effectiveness of the RHM against more intelligent attacks such as advanced persistent threats (APTs). Similarly, Jafarian et al. [76] presented a spatiotemporal address mutation technique that aims to mitigate the impact of APTs by varying host IP binding dynamically based on location and time. In addition, Jafarian et al. [74] investigated the effectiveness of the address mutation approaches [3] [73]. However, their effectiveness evaluations are limited to the effects of the scanning

worm's attacks. MacFarland and Shue [103] used a host IP address mutation to defend a large-scale network by employing an SDN controller that controls DNS interactions. Antonatos et al. [10] proposed an IP randomization technique to thwart hitlist worms attacks, aiming to avoid malicious worms that gather information about victim targets in a networked system and making it hard for attackers to identify new, vulnerable targets. Jafarian et al. [73] implemented an IP shuffling technique mutating the IP addresses unpredictably while minimizing the overhead of the MTD operations. The authors used the SDN-based OpenFlow (OF) protocol that frequently assigns Virtual IP addresses translated from and/or to a real host's IP address.Carroll et al. [23] also studied network address shuffling to protect honeypots as an MTD technique based on probability models to quantify the attack success probability with respect to the size of a network, the number of addresses scanned by attackers, the number of system vulnerabilities, and the frequency of triggering the shuffling-based MTD. Kampanakis et al. [81] employed host randomization and mutation, evaluating the performance in terms of the attackers' overhead increased by the MTD technique used in an SDN platform.

- **Port Hopping**: Lee and Thing [93] proposed a port hopping technique for mitigating DoS attacks. It changes a server's port numbers dynamically based on time, and a cryptographic key shared between server and client, which adds overhead for time synchronization and key management.Shi et al. [143] proposed a port and address hopping technique for active cyber defense. However, it is a timestamp-based synchronization hopping of services, and its scope is limited to metropolitan area networks. Luo et al. [99] proposed a random hopping-based MTD changing the port numbers and IP addresses. It maps IP and port in a one-to-one manner; however, it is only applied to conventional

networks requiring additional overhead to maintain the infrastructures. Zhang et al. [169] proposed a port hopping technique to mitigate DoS attacks in SDN. The proposed technique takes a timestamp feedback-based synchronization approach while requiring extra ports for listening and receiving the feedback messages, resulting in increasing of the attack surface. Luo et al. [98] studied a port hopping technique based on the concept of shuffling-based MTD to effectively deal with reconnaissance attacks by hiding service identities while increasing confusion for potential attackers. This work quantifies the effectiveness of the proposed port hopping technique based on the attack success probability as a function of the values of key design parameters, including the size of a port pool, the number of probes, the number of vulnerable services, and the frequency of port hopping.

- **Packet/Frame Header Randomization**: Wang et al. [158] proposed a technique called U-TRI, Unlinkability Through Random Identifier, which uses the randomization of a packet header identifier to confuse attackers. Skowyra et al. [144] proposed a packet header randomization (PHEAR) technique that eliminated implicit and explicit network identifiers from a packet header. However, PHEAR was designed to be used only within internal networks, relying on existing security mechanisms (i.e., IKE2 and IPSec).

- **Network Topology Shuffling**: The underlying idea of this technique is to invalidate an attacker's path information by continuously changing routes in networks. Achleitner et al. [2, 1] proposed a virtual topology generation framework against network scanning attacks by leveraging the SDN technology. Hong et al. [65] presented an optimal network reconfiguration technique based on the concept of shuffling-based MTD for SDN environments. They solved a shuffling assignment problem and showed the increase in network security when network routing paths are

diversified.

- **VM / Proxy Migration**: Danev et al. [41] considered securely migrating Virtual Machines (VMs) as an Msome emulationTD mechanism in a private cloud. The underlying idea of this work is to utilize an extra physical Trusted Platform Module (TPM) and trusted parties for the migration process as well as public key infrastructure to secure the protocol. Zhang et al. [170] developed a periodic VM migration as an MTD technique based on the balance between the level of security obtained and the cost incurred upon the migration of VMs. Penner and Guirguis [123] developed a set of MTD technologies to change the location of VMs in a cloud to defend against attacks leveraging a Multi-Armed Bandit (MAB) policy, which aims to exploit VMs providing the highest reward. The proposed MTD techniques are deployed to minimize the MAB attacks launched by the attacker aiming to obtain credentials (e.g., credit card information) or critical data. The proposed MTD techniques are evaluated based on how much time it takes to switch VMs. However, the enhanced security introduced by the MTD techniques has not been investigated. Jia et al. [79] devised an MTD technique to deal with Distributed Denial-of-Service (DDoS) attacks through securing data transmission between authenticated (legitimate) clients and a protected server. This method used the technique called *client-to-proxy shuffling* that continuously moves secret proxies. For evaluation, they assessed the resistance of their method toward brute-force attacks, as well as proxy-based and communication-based overhead. Peng et al. [122] proposed VM migration or snapshotting and diversity or compatibility of migration for cloud networks. This work considered attackers' learning based on accumulated intelligence obtained and dynamics and heterogeneity of a service's attack surface. Finally, this work proposed a probability model to propose an MTD service deployment strategy. This study found that

the proposed MTD technique is more effective under a dense service deployment and strong attacks while the defender's awareness toward the heterogeneity and dynamics of the attack surface is helpful for enhancing the effectiveness of the MTD as it can be used to determine when/how to perform the MTD.

- **Software / Service Reconfiguration**: Casola et al. [26] developed an MTD technique based on the reconfiguration of devices in an IoT environment whose cryptosystems and firmware versions are shuffled. Vikram et al. [155] proposed a shuffling based MTD technique on the application layer for a secure web through randomizing HTML elements. Most bots attacking the web use static HTML elements in the HTTP content/form page. Hence, the randomization of those HTML elements and parameters could be an appropriate technique for avoiding web bot attacks.

- **Platform Diversity**: Okhravi et al. [116] proposed the Trusted Dynamic Logical Heterogeneity System (TALENT) framework for live-migrating of the critical infrastructure applications across heterogeneous platforms, which permits running of the critical applications to change its hardware and operating system (OS). It changes a platform on-the-fly, creates a cyber moving target, and provides the cyber survivability through platform diversity. TALENT creates a virtualized environment at the OS-level using containers with a checkpoint compiler for migrating a running application of different platforms. It preserves the state of the application, such as the execution state, the open files, and the network connections during the migration.

- **OS-Rotation**: Thompson et al. [151] developed the Multiple OS Rotational Environment (MORE) MTD based on the existing technology to achieve a feasible MTD solution at an OS-level. MORE MTD consists of a set of virtual machines (VMs) equipped with a different distribution

of OS (i.e., different Linux distributions) and web applications. The periodic rotation of the hosts/VMs creates a dynamic environment controlled by an administrator's machine running a daemon process. This technique reduces the likelihood of successful exploitation of the OS's vulnerabilities and its security impact by limiting the duration of the rotation window. The "rotation window" is the duration of an OS being exposed and vulnerable to an attack.

Shuffling-based MTD techniques can easily work with existing technologies without developing another security mechanism that requires a thorough security analysis. Hence, in terms of the development cost and resources, these methods are useful, immediately applicable, and highly compatible with legacy devices and technologies. However, since shuffling relies on the quality of existing technologies, if those existing technologies are not robust enough against attacks (i.e., well-known vulnerabilities to attackers), the security achieved by this shuffling technique can be significantly limited by those vulnerabilities of the legacy devices/technologies. In this case, the shuffling frequency can be increased to help cover those vulnerabilities, but this also will increase the defense cost incurred by over-utilizing MTD as another layer of defense. Furthermore, the shuffling spaces (e.g., many virtual IPs that can be assigned for a real IP) are critical to enhancing security.

2.2.2.2 Diversity

This technique employs the deployment of system components with different implementations that provide the same functionalities. The examples include the use of diverse paths for routing or the change of platforms consisting of different implementation of software components or migration between different platforms (i.e., software stacks or hardware). Diversity-based MTD aims to enhance system resilience by increasing fault-tolerance in that the system can provide normal services in the presence of attackers in the system. In this section, we discuss how system diversity is realized based on a variety of

domains.

- **Software Stack Diversity**: Huang and Ghosh [68] and Huang et al. [69] introduced a diversity MTD technique in order to enhance network resilience and service provisions. They deployed a diversity method to the virtual servers (VS) like OSs, virtualization components, web servers, and application software. They evaluated the proposed diversity method with respect to attack success probability.

- **Code Diversity**: Azab et al. [14] developed an MTD technique that changes a running program's variants erratically based on the concept of diversity. The proposed method divides a large program into components (e.g., cells or tasks) that can be performed by functionally-equivalent variants. Their method includes a recovery mechanism to enhance the resilience of the proposed technique. Choosing a different variant at runtime makes it hard for an attacker to penetrate and scan the system. The proposed method can mitigate even the effect of the attack success by affecting one variant that can be instantly replaced by another variant through the recovery mechanism. However, the robustness against attacks based on security analysis is not clear, while its implementation is quite complex, which can be questioned for applicability in diverse contexts. In automated software diversity, a program can diversify at instruction, block, loop, program, and system levels in different phases of the life cycle, such as compilation and linking, or installation [92]. A JIT (just-in-time) compiler (i.e., Java/Python JIT compilers) generates code during the execution of the program. The JIT compilers can be used for the code diversification. In this case, the security of the system depends on how predictable the JIT compilers are. The code diversity can be enhanced with adaptive notification-oriented programming (i.e., assembly language instruction, no-op) [78].

- **Programming Language Diversity**: Taguinod et al. [148] focused on an MTD technique based on the diversification of programming languages to avoid code and SQL injection attacks. This work proposed the MTD technique to be applied in different layers of the web application to change the implemented language of the web application without affecting or disrupting the system functionality.

- **Network Diversity**: Zhuang et al. [174] investigated the relationship between the diversity of network configurations and attack success probability by evaluating the MTD technique based on a logical mission model. The authors examined how the network size, the frequency of shuffling / adaptations, and the number of vulnerable computers affect the effectiveness of the MTD technique based on the experimental results tested from the network security simulator, `NeSSi2`.

Like shuffling-based MTD techniques, diversity-based MTDs can leverage the existing technologies. As discussed earlier, diversity-based MTD techniques are often combined with shuffling-based counterparts to double the effectiveness of a proposed MTD. However, since the diversity-based MTD is also based on the existing technologies, if those technologies have high vulnerabilities to attacks, introducing MTD as another layer of defense does not provide better security, even with the sacrifice of additional defense costs. Also, the degree of diversity significantly affects the effectiveness of the MTD. For example, if there are not many alternatives to use (e.g., only two versions of software that provides the same functionality with different implementations), there will be inherent limitations in enhancing security.

2.2.2.3 Redundancy

This technique provides multiple replicas of system (or network) components, such as multiple paths between nodes in a network layer or multiple software components providing the same functionality at the

application layer. The key aim of redundancy-based MTD is to increase system dependability (e.g., reliability or availability) by providing redundant ways of providing the same services when some of the network nodes or system components are compromised. In this sense, redundancy contributes to increasing system (or network) resilience in the presence of insider threats. This technique can often be combined with diversity-based MTD so that, for example, redundant services are available where the attackers are required to know additional credentials or intelligence to use other alternative components (e.g., different or additional credentials or a different level of privileges). In this section, we discuss how these techniques are studied in the existing approaches.

- **Redundancy of Software Components**: Yuan et al. [165] proposed a redundancy method for web servers to prevent malicious code injection attacks on a web server by developing a self-protection model, including architectural adaptation threat detection and mitigation. They used the so-called *agreement-based redundancy* that provides replicas of software components at runtime. However, this work did not investigate the effectiveness of the proposed MTD mechanism.

- **Redundancy of Network Sessions**: Li et al. [95] proposed a traffic morphing mechanism by adopting the concept of MTD in terms of redundancy for cyber-physical system (CPS) environments. The proposed traffic morphing algorithm is designed to protect CPS sessions by maintaining several redundant network sessions that have the distributions of inter-packet delays indistinguishable from those observed in normal network sessions. A CPS message will be disseminated via one of these sessions to meet its given time constraint. In the process of dynamically adjusting the morphing process, this work showed the low complexity of the proposed work and high adaptivity to the dynamics of CPS sessions.

Unlike shuffling or diversity-based MTDs, redundancy-based MTD has a

greater bearing on service available to users, which is often measured by some concept of system dependability (e.g., availability or reliability) in which the availability (or reliability) can be easily interrupted by performing shuffling or diversity-based MTD, particularly incurring interrupted services. Note that redundancy-based MTD can be well-mingled with the other two techniques to increase both security and performance. However, creating additional replicas of system components (e.g., servers or paths) incurs an extra cost. Also, if redundancy-based MTD is not properly executed, it provides an even greater chance for an attacker to perform attacks on a larger attack surface (i.e., another server to attack or another path to reach a target) than the system without the redundancy-based MTD.

2.2.3 Discussions on MTD Classifications

While the timeliness-based MTD categorizes MTD techniques based on the principle of 'when to move,' the operation-based MTD classifies MTD techniques by answering 'how to move.' In the timeliness-based MTD, when a defender has high uncertainty about an attacker's current activities and/or behaviors, it is better to trigger an MTD operation based on a certain, fixed time interval. However, after the defender becomes certain about the attack patterns/activities, it can detect the system's vulnerabilities more accurately. And then, the defender can adaptively execute the MTD operation to save defense costs by using an event-based MTD, for example, triggering an MTD operation based on intrusion alerts.

Shuffling, diversity, and redundancy are operations or mechanisms which are based on 'how to move' key design principle of MTD. Although these three types of MTD operations can be used to classify MTD techniques, they often support each other directly or indirectly in realizing their respective goals. We summarize the connections between the SDR types and their goals in Fig. 2.4. First, the concept of *diversity* is adopted from the principle that the diversity of system components can enhance security (i.e., software polyculture enhances

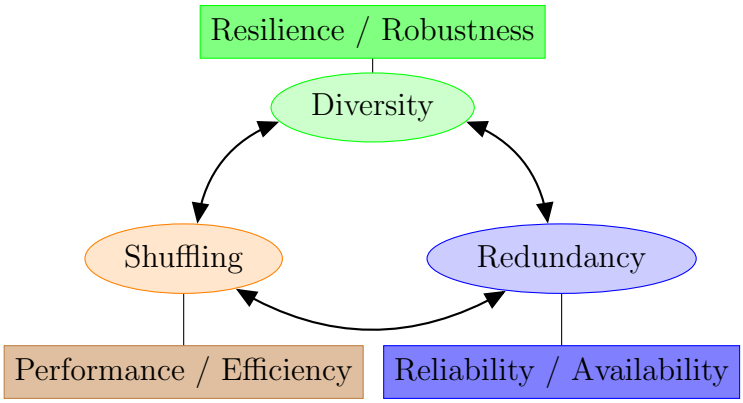


Figure 2.4: Relationships between shuffling, diversity and redundancy.

security [51, 55, 149]), leading to high system resilience (or robustness) even in the presence of attacks. If system diversity is high, there exists a variety of alternatives to the provision of the same service, such as multiple routes from a source to a destination to deliver a message.

Second, *shuffling* also can enhance diversity because how to rearrange system components can affect the degree of system diversity. For example, a platform migration using a shuffling technique leads to higher diversity. How to shuffle targets is closely related to how much cost incurs from shuffling the targets and, accordingly, how much it can enhance security or interrupt services to normal users. The effectiveness of shuffling is also affected by how diverse a target is in nature. For example, given a network, we want to increase the number of different types of software that provides the same set of services. However, if we only have one or two software types, then there is an inherent limitation of the performance achievable by the shuffling operation. On the other hand, if the shuffling is optimally performed, system diversity also increases, leading to higher system resilience against attacks.

Finally, high diversity and effective shuffling are critical to increasing redundancy for achieving higher reliability (or availability) while properly dealing with attackers. Particularly, if redundancy is high with low diversity due to a poor or no shuffling technique, it can also be leveraged by the attackers to use alternate ways to get into the system persistently. Therefore, these three

types of techniques, shuffling, diversity, and redundancy, namely SDR, impact each other and should be considered in combination to enhance system security or resilience properly.

These two classification methods answer two different design questions in developing MTD techniques (i.e., when-to-move and how-to-move). Hence, we can even consider both timeliness-based and operation-based classifications in order to develop a secure, affordable, adaptive, and QoS-aware MTD solutions.

2.3 Attacks Modeling in MTD

In this section, we discuss the characteristics of attacks considered in the state-of-the-art MTD techniques. We discuss (1) the characteristics of advanced attacks in Section 2.3.1; (2) the cyber kill chain model in Section 2.3.2; (3) attack types mainly considered in the existing MTD approaches in Section 2.3.3; and (4) discussion on the limitations of existing attack models in Section 2.3.4.

2.3.1 Characteristics of Advanced Attacks

MTD techniques have been developed in order to deal with increasingly more sophisticated, intelligent, and persistent attacks equipped with more advanced tools. We discuss the key characteristics of these advanced attackers as follows:

- **Persistent attackers**: In the existing MTD works, we found that attackers are *persistent*, not performing an one-time attack (e.g., APT attacks [18, 43, 48, 67]). This persistent attack behavior is well observed in multi-stage attacks that start from scanning attacks in the reconnaissance stage before attacks obtaining access into a system (i.e., outside attackers) and continue to the attack delivery or exploitation stage after they break into the system (i.e., inside attackers).

- **Adaptive attackers**: Attackers are adaptive to dynamically changing system conditions and external environmental conditions, as they take into consideration both physical and cyber accessibility. These attackers also have intelligence with regard to their resources, executing adaptive attacks [80, 156, 162] that wisely manage their resource limits and at the same time opportunistically seek to compromise an entire system.

- **Stealthy attackers**: Attackers do not exhibit an identifiable attacking behavior all the time. They perform attacks in a highly stealthy manner [3, 73], even showing well-behaved features of good citizens. However, at a time when an attack is calculated to inflict serious harm or damage to the system, they exhibit attack behaviors. However, they stay stealthy until the time comes.

- **Incentive-driven attackers**: MTD is developed to deal with smart attackers. In particular, attackers are intelligent enough to execute attacks efficiently, such that the attack has a minimal cost but maximal outcome. Therefore, we can consider the attacker to be a rational actor that is sensitive to incentives, such as attack success with minimum cost [50].

As a typical type of sophisticated attacks, the *Advanced Persistent Threat* (APT) has been commonly considered as an attack-type techniques need to countermeasure [18, 43, 48, 67, 128, 172]. More or less, APT attackers show the above features in performing their attacks and are often described under the scenario of the Cyber Kill Chains [70]. In the next section, we discuss an overview of the cyber kill chain and the behavior of an APT attack at each stage of the chain to understand how MTD techniques need to consider APT attacks.

2.3.2 Cyber Kill Chain

A new class of adversaries with sufficient resources and high intelligence has emerged, which is the so-called 'Advanced Persistent Threat (APT),' as mentioned in Section 2.3.1. APT is capable of multi-year intrusion campaigns to obtain highly sensitive information, such as proprietary corporate information or national security secrets [70]. In order to defend against APTs, an intelligence-based defense model is critical to defenders for mitigating the risk and vulnerability of attacks to a system. Hutchins et al. [70] developed the intelligence-based defense model, called the *Cyber Kill Chain* (CKC), based on multiple phases of a cyber attack. The CKC model provides: (1) the description of intrusion phases; (2) mapping the indicators of adversary kill chain to the courses of defense actions; (3) the pattern identification associating individual intrusions with broader campaigns; and (4) the understanding of iterative intelligence gathering. The CKC model consists of the following phases [70]:

1. *Reconnaissance*: This phase involves researching, identifying, or selecting targets, often by crawling Internet websites to obtain target information, such as email addresses, social relationships, or information regarding particular technologies the target system uses.

2. *Weaponization*: This phase uses a remote access trojan to be coupled with an exploit into a deliverable payload (e.g., Adobe Portable Document Format or Microsoft Office documents) using an automated tool, called a *weaponizer*.

3. *Delivery*: In this phase, the weapon is transmitted to a targeted system using delivery vectors for weaponized payload by APT actors (e.g., email attachments, website, and USB removable media).

4. *Exploitation*: After the delivery of the weapon to a victim host, the intruders' code is triggered by exploitation, commonly targeting for an application or OS vulnerabilities.

5. *Installation*: The adversary can stay inside the target environment by installing a remote access trojan or backdoor on the target (or victim) system.

6. *Command & Control* (C2): Compromised hosts try to establish a C2 channel by beaconing outbound to an Internet controller server. Upon the establishment of the C2 channel, the intruders can take control of the target system (e.g., 'hands on the keyboard').

7. *Actions on Objectives*: After the previous six phases are successfully performed, adversaries launch attacks to breach security goals (e.g., data integrity, availability, confidentiality).

Okhravi et al. [117], and Ward et al. [159] described CKC model in a shorter version with the phases of reconnaissance, access, exploit development, attack launch, and persistence in which the 'access' phase incorporates the 'weaponization' and 'delivery' phases from [70] and the 'attack launch' phase incorporates 'installation' and 'C2' [70] as well.

2.3.3 Attacks Considered in MTDs

In this section, we discuss common types of cybersecurity attacks considered and mitigated in the existing MTD approaches.

- **Reconnaissance (or scanning) attacks**: Scanning attacks are used by attackers to gather information and intelligence about a target system before an actual attack is launched. The attackers usually use a customized set of software tools (e.g., *Nmap* [101]) to scan the target system (or network) to find information, such as OS types, IP addresses, port numbers, running services, protocols, network topology, and exploitable vulnerabilities. An attacker/scanner performs scanning by sending probe packets (e.g., ICMP echo request message or TCP SYN, TCP ACK, TCP Xmas, UDP port scanning, etc.) to the target host(s)

in the networks. To counter reconnaissance attacks, several network address-based approaches have been proposed [74, 3, 2, 3, 23, 44, 73, 85, 99]. Similarly, MT6D [45] provides a mechanism to thwart scanning attacks in the IPv6 environment by dynamically changing IP addresses to invalidate scan results.

- **DoS (or DDoS) attacks**: A DDoS attack uses a large number of botnets (i.e., a set of bots, consisting of compromised machines, or zombie machines) to attack a targeted system by flooding traffic messages (e.g., UDP floods, ICMP floods, and/or SYN floods) from multiple sources to force it to shut down or to deny services to legitimate users. DDoS often involve time-based attacks that can cause targeted systems to miss critical deadlines for updates or to fail to complete critical tasks. Common MTD approaches to handle these attacks include the use of hidden proxies, IP/port shuffling and/or address mapping [13, 32, 52, 59, 80, 79, 85, 93, 143, 147, 162, 109]. For example, Meier et al. [109] proposed a novel network obfuscation approach and implemented a framework named 'NetHide' which can successfully battle against the possible attacks such as Link-Flooding Attacks (LFAs) launched by even advanced attackers. The MTD strategy behind the NetHide is to change and modify path tracing probes in the data plan. They showed that NetHide could hold a trade-off between security (i.e., creating difficulties for attackers) and usability (e.g., high accuracy and low network functionality degradation). Their results showed that NetHide could mitigate the probability of attack success by 1% while it provides 90% and 72% accuracy and utility, respectively. Moreover, MOTAG [79] mitigates the flooding DDoS attacks using hidden proxies as moving targets to secure service access for authenticated clients.

- **Buffer overflow attacks**: This kind of attack usually happens due to a lack of buffer boundary checking. Common memory protection

techniques, such as Address Space-Layout Randomization (ASLR) [137], and Address Space Layout Permutation (ASLP) [86], are proposed to thwart the buffer overflow attack. These techniques randomize the memory position of data and program segment, library, and so forth. Manadhata [104] and Manadhata and Wing [106] also considered a buffer overflow, leveraging the attack surface of a defense system and proposed a countermeasure against it by reducing or shifting the attack surface in terms of the system's methods, channels, or data items.

- **Worm attacks**: A worm is malicious software (malware) that replicates itself and employs a network to spread itself to other machines by relying on existing bugs or holes in the target system. The worms cause damages, such as consuming bandwidth or turning a worm-infected computer into a botnet that can be used by the worm's programmer to profit by sending spam or conducting DDoS attacks. A countermeasure against worm attacks is Network Address Space Randomization (NASR) [3, 10, 76] that randomizes the host IP address (i.e., IP shuffling).

- **APT attacks**: APT attacks are well known as one of the sophisticated attack types aiming to access system resources, control them, and perform exfiltration attacks (e.g., leaking sensitive or confidential information out to unauthorized parties) by performing stealthy, persistent, and adaptive attacks [2, 9, 24, 25, 29, 50]. The behaviors of APT attackers can be observed in the cyber kill chain, which is described in Section 2.3.2. APT attacks are also called 'multi-stage attacks' that refer to attackers performing multiple attacks across multiple stages. For example, the attacks range from network scanning and packet sniffing to illegitimate authentication and service interruption (e.g., Stuxnet virus) [18, 43, 48, 128, 172].

- **Side-channel attacks on VMs**: Attackers perform attacks over side channels based on the shared CPU cache of a VM in order to obtain

sensitive information [170].

- **Attacks on Web applications**: Diverse types of attacks can be performed on web applications, such as SQL injection, directory traversal, and cross-site scripting, which breach the key security goals that are confidentiality, integrity, and/or availability. These types of attacks can be countermeasured by software stack diversity and/or redundancy [112, 135, 165].

We summarized the main attack types considered in existing MTD approaches such as *shuffling*, *diversity* and *redundancy* with 36 researches (i.e., papers) published for 2010-2018 in Fig. 2.5, Fig. 2.6 and Fig. 2.7 respectively. Note that more than one technique or attack can be considered in each work. The three major attacks the existing MTD techniques have mainly addressed are reconnaissance attacks, DoS attacks, and APT attacks. Also, most proposed approaches are shuffling-based MTD while diversity-based MTD techniques are often combined with the shuffling-based techniques. But redundancy-based techniques are rarely discussed to deal with the attack behaviors addressed above because the redundancy-based approaches have been studied in the dependability domain (e.g., reliability, availability) rather than for MTD and often used in combination with either shuffling-based or diversity-based MTD techniques.

2.3.4 Discussion on Existing Attack Models of MTDs

In this section, we discuss on limitations of the existing attack models considered in MTD based on attacker types and attack behaviors described in Section 2.3.1 and Section 2.3.3.

- *Smart and intelligent attackers are significantly less considered:* Most MTD techniques are shuffling-based to deal with attacks. If attackers are intelligent such that they can detect defense patterns by using learning

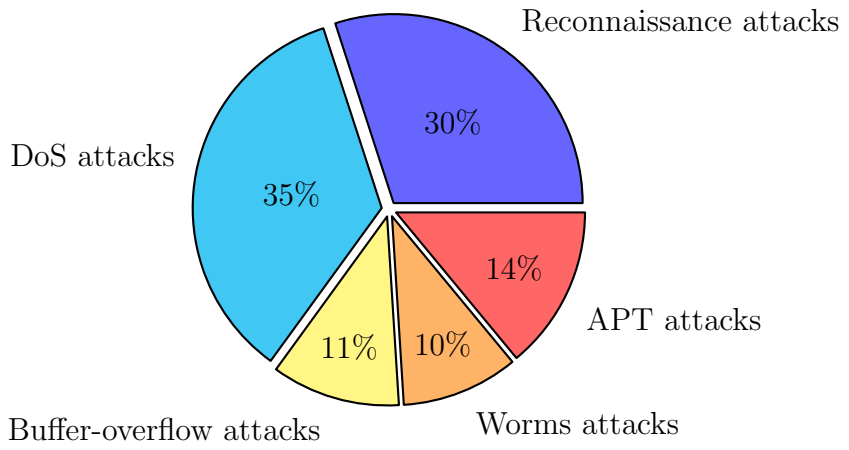


Figure 2.5: Attack types considered in shuffling-based MTDs.

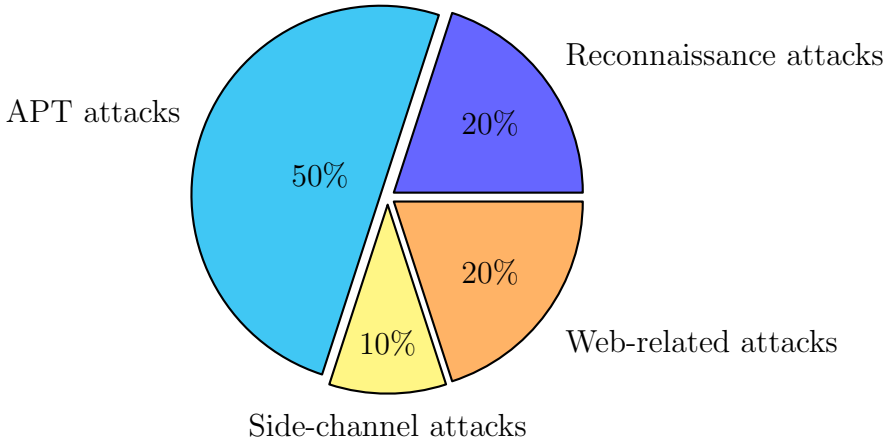


Figure 2.6: Attack types considered in diversity-based MTDs.

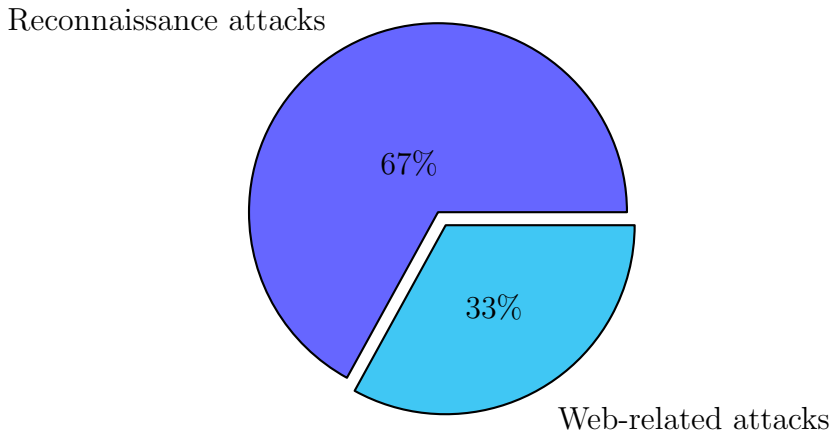


Figure 2.7: Attack types considered in redundancy-based MTDs.

mechanisms (e.g., machine learning or cognitive learning), then they can easily capture what types of MTD techniques are used and what the patterns are to trigger an MTD operation. However, highly intelligent, learning attackers are not considered. Generally, it is assumed that attackers have certain attack patterns, rather than that they learn and can launch adaptive attacks. Whereas the contrary is true for defenders, they are often permitted to take proactive, adaptive defense actions based on an assumed highly intelligent learning capability.

- *Few scenarios are considered to deal with multiple strategies by attackers and defenders:* A specific attack type can be easily mitigated or prevented by a particular MTD technique. However, if an attacker identifies an exploit that is not covered by the deployed MTD technique, the attacker can even use the exploit and successfully launch its attack to penetrate the system. Unfortunately, most MTD approaches focus on a single or a small set of attacks, and few scenarios have considered multiple strategies that can be taken by attackers or defenders. For example, the attacker may need to decide which system vulnerability it needs to target while the defender may want to determine which defense mechanism needs to be used, where both parties aim to choose a cost-effective decision under resource-constrained, time-sensitive settings.

- *An attacker has been less considered as a rational decision-maker with learning ability:* The smart attackers can leverage how MTD works and what side-effect the MTD can introduce. Also, they can capture the defense patterns for an adaptive MTD, which aims to realize a cost-effective MTD. However, attackers with learning capability and their mental models have been rarely studied while high intelligence with the defender and corresponding intelligent MTD (e.g., machine learning-based MTD) is easily assumed and has been proposed [33, 152, 155, 171].

2.4 Metrics for Assessing MTDs

In this section, we discuss what existing metrics have been proposed to assess the effectiveness and efficiency of the existing MTD techniques. Along with the discussions of the overall trends observed from this literature review, we discuss some limitations of the metrics used in the state-of-the-art MTD approaches.

2.4.1 Metrics for Measuring Effectiveness of MTDs

Diverse security assessment models and metrics have been proposed for measuring the effectiveness of the MTD techniques. Zaffarano et al. [166] proposed a quantitative framework measuring productivity, success, confidentiality, and integrity for given missions and attacks models. Taylor et al. [150] performed a series of experiments to assess the effectiveness of network-based MTDs using the metrics in [166]. However, the usage of the models is limited to evaluating network-based MTDs, such as ARCSYNE (Active Repositioning in Cyberspace for SYNchronized Evasion), and SDNA (Self-shielding Dynamic Network Architecture) [164]. Carroll et al. [23] proposed a probabilistic model and analyzed for the performance of the network address shuffling MTD technique. However, the model is designed only for the assessment of address shuffling-based MTD technique. Several entropy-based approaches [177, 102] have been proposed for the quantitative security assessment of MTD techniques. However, the accuracy of the entropy-based metrics is based on probability distribution measuring the likelihood of happening the event.

We study the existing metrics to measure the effectiveness of MTD techniques in terms of the perspectives of an attacker and a defender. The attacker's metric estimates its attack performance, indicating that the attacker's high performance refers to the defender's low performance, and vice-versa. The defender's metric measures its performance in achieving the security

and/or defense goals of a given system. The *attacker's metrics* estimate the adverse impact of the proposed MTD techniques on the attacker's performance and the metrics are obtained by:

- **Attack success probability (ASP)** [3, 9, 18, 23, 24, 26, 29, 43, 48, 130, 166]: This metric refers to the probability that attacks are successfully performed. For example, it refers to the probability that a system component (or defender) is compromised or a target is successfully discovered and/or accessed by an attacker. Rahman et al. [130] also used a metric called *attackability*, which refers to the probability that an attacker can access system states (or components) to the attack. This metric combines the degree of system vulnerability plus the feasibility of an attacker accesses and performs an attack based on its resource level. However, in the sense that it measures the degree for the attacker to successfully access a system component, this metric is aligned with ASP.

- **Attack utility** [5, 6, 18, 112, 128, 172, 173]: When the interactions between attackers and defenders and their best strategies are considered based on game-theoretic approaches, the payoff (or utility) of an attacker is used to measure the gain or loss by deploying a proposed MTD.

- **Learning by attackers** [172]: This measures the degree of an attacker's leaning toward the payoff obtained by a defender upon the performed attack.

- **Mean time to compromise a system (MTTC)** [4, 24, 25, 29, 176]: This indicates how long an attacker takes to compromise an entire system. In terms of a defender's perspective, this metric is similar to *mean time to failure* (MTTF), as described under a defender's metrics below.

- **Unpredictability** [58, 103, 138]: This indicates how much confusion and/or uncertainty a given MTD has introduced to attackers.

- **Attack surface** [106, 105]: This metric is defined as the number of system resources that can be used by attackers to attack the system, such as channels, data items, and/or methods. Hence, a larger attack surface exposes more vulnerabilities.

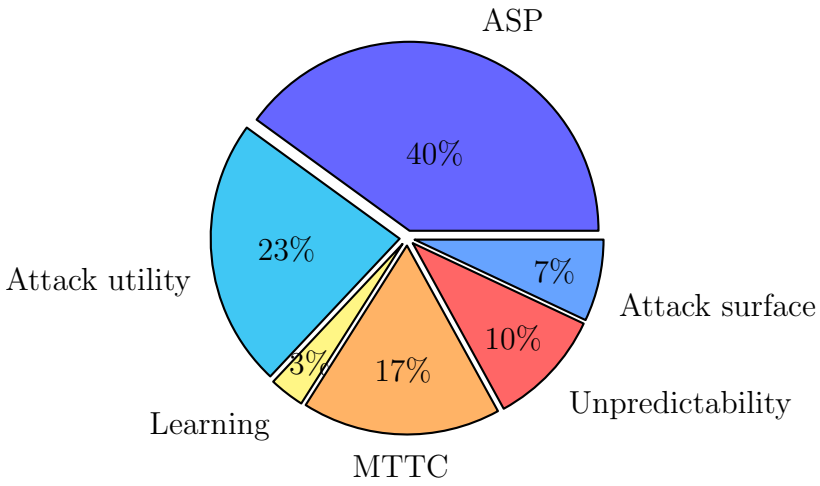


Figure 2.8: Metrics measuring MTD effectiveness by an attacker's perspective.

In the literature, various types of the *defender's metrics* are used to measure MTD effectiveness in terms of the following metrics:

- **Defense success probability (DSP)** [3, 31, 33, 166]: We call metrics measuring the success of an MTD technique DSP in this work. Zaffarano et al. [166] estimated the rate of executing successful defenses (e.g., for a defender, the rate at which tasks are executed and completed) or attacks (e.g., for an attacker, the rate at which attacks are performed and successfully completed). Colbaugh and Glass [33] used a detection accuracy of anomaly behaviors, such as attack behaviors or spams, in order to determine whether to trigger an MTD operation. Clark et al. [31] measured the portion of decoy nodes detected by attackers when IP randomization techniques are used in order to measure the success of the IP-shuffling MTD strategy. Al-Shaer et al. [3] measured an IP-mutation success probability to measure the effectiveness of the MTD. This metric measures the probability that a mutated IP is not hit by scanning attacks.

- **Mean time to failure (MTTF)** [24, 25, 29, 176]: This refers to a system reliability metric capturing the system's up-time in the presence of attacks when failures can happen due to either defects or security threats. This metric is the same as MTTC under the attacker's metrics.

- **Defense utility** [18, 112, 128, 172, 173]: Game-theoretic approaches for the optimal deployment of MTD techniques have taken to identify the best defense strategy by a defender. The payoff (or utility) of the defender measures the effectiveness of an MTD technique.

- **Learning by defenders** [172]: This metric measures the degree of a defender's leaning toward the payoff an attacker has obtained upon a defense action taken by the defender.

- **System security**: Various kinds of metrics measure the system security properties enhanced by proposed MTD techniques:

  - **Confidentiality** [128, 170, 166]: This measures how many system components are compromised [128]. In some contexts, some information should be kept confidential, such as private information. The degree of preserving confidential or private information is another metric to indicate the degree of security. In [166], mission confidentiality refers to the degree of exposing confidential information to unauthorized parties while attack confidentiality means the degree of attack behaviors detected by a defender.

  - **Integrity** [166]: Integrity metric is discussed in terms of mission integrity and attack integrity. Mission integrity refers to how much information related to executing a given mission is communicated without being modified and/or forged, while attack integrity indicates how much accurate information the attackers' view.

  - **Availability** [58, 128]: This indicates the portion of system assets that are not compromised to provide a normal service.

  - **Degree of vulnerability** [4, 5, 6, 18, 23, 24, 25, 172]: This measures the probability that a given platform to be selected is vulnerable during a particular period, or a given system component is vulnerable because an attacker controls it.

- **Other metrics**: Based on the unique features of each of the existing MTD approaches, various other types of metrics have been adopted to measure the effectiveness of MTD as follows:

  - **Controllability** [128]: This refers to the portion of critical system assets that expose a high vulnerability to an attacker if compromised.

  - **Worm propagation speed** [3, 74, 73]: This measures how much a deployed MTD can slow down actions by an attacker. This also indirectly increases the detection of attackers by earning more time to monitor the attacker.

  - **Vastness** [58, 103, 105, 170]: This measures the size of spaces that a given defense mechanism can cover, such as IP spaces an attacker needs to scan through. Also, the number of target hosts set by a defender can consume an attacker's resource because it determines all the possibilities the attacker needs to scan through. In [105], this is considered based on the metric called 'attack surface measurements.'

  - **Periodicity** [58, 103]: This estimates how often system configurations change in order to provide a sufficient level of confusion to attackers.

  - **Uniqueness** [58, 103]: This measures how uniquely an individual entity (e.g., a host) is authorized to a system without being accessed by other entities.

  - **Revocability** [58, 103]: This measures the degree of frequency to

terminate or expire a prior system configuration (e.g., access control or a given IP configuration).

- **Distinguishability** [58, 103]: This measures how well a given defense distinguishes trustworthy entities from non-trustworthy entities.
- **Loss in rewards between an optimal deployment and an executed deployment** [135]: This metric captures how much loss occurred for the actual execution of an MTD operation over the optimal deployment.

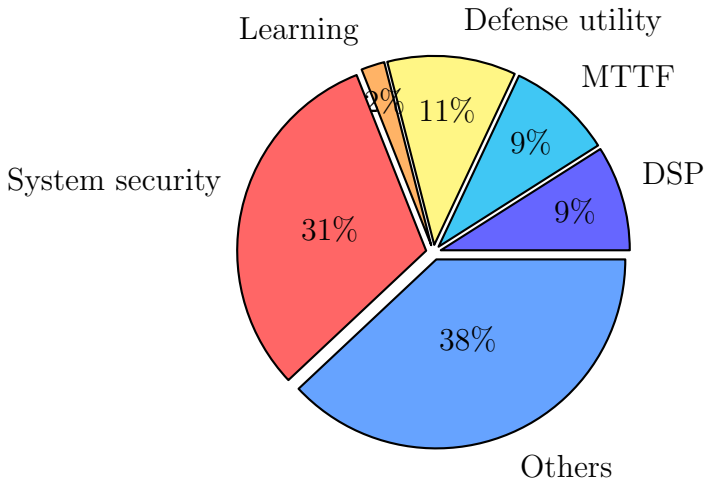


Figure 2.9: Metrics measuring the effectiveness of the MTDs by a defender's perspective.

The metrics to measure the effectiveness of existing MTD techniques are summarized in Figs. 2.8 and 2.9 based on 27 papers published during 2011-2018.

2.4.2 Metrics for Measuring Efficiency of the MTDs

The **attacker's metrics** are used to capture how much cost (or penalty) is introduced for an attacker to achieve attack success when a proposed MTD is deployed, as follows:

- **Penalty in attack payoff** [50, 79, 162, 170]: Many game theoretic MTD approaches estimate attack cost at an abstract level (e.g., the cost is 1 for attacking; 0 otherwise).

- **Attack cost** [6, 5, 81, 104]: This measures how much overhead and/or impact is introduced to attackers to perform their attacks. To be specific, an attacker's scanning tool, such as $Nmap$, is used to capture the scanning overhead [81].

The MTD efficiency by the attacker's perspective is mainly measured by two types of metrics as above, although both metrics are concerned about resources the attacker needs to invest to launch a planned attack. Due to the similar nature of both metrics and their small volume, we omit the figure.

The **defender's metrics to measure MTD efficiency** that are commonly observed in the literature include:

- **Quality-of-Service (QoS) to users** [31, 60, 162]: This metric captures the degree of service quality provided to users while implementing a given MTD technique as triggering an MTD (e.g., platform migration) often hinders service availability to normal users [60, 162]. Also, upon deploying IP mutation techniques, the number of connections interrupted [31] is used to measure QoS provided to users.

- **System performance** [7, 30, 32, 45, 52, 60, 72, 95, 110, 127, 155, 156, 168]: This metric measures how much overhead is introduced to deploy a given MTD, such as message overhead (e.g., delay, packet loss, or control packet overhead) [45, 168], operational delay [110, 168] / cost [52, 60, 155, 156, 168] to deploy an MTD, the number of dropped connections [32], or performance overhead (e.g., file sizes or performance degradation to distribute software for diversity) [72]. System performance is also captured by system throughput measuring how much a given system (or network) can maintain its performance in terms of network

throughput (e.g., how many messages are correctly delivered) [7, 168] or server throughput (e.g., how many queries are properly provided) [127].

- **Defense cost**: Various aspects of defense cost are measured to indicate the efficiency of MTD techniques. An abstract level of defense cost (i.e., migration cost or maintenance cost of VMs) [29, 50, 79, 162, 170] is used to measure the cost of a deployed MTD mechanism, which is mostly used in game-theoretic MTD. Some defense cost captures the level of infrastructure (e.g., several numbers of proxies or decoy nodes) required to ensure a required level of service availability [79, 162]. Some other works also used specific metrics to capture defense cost as follows:
  - **Address space overhead** [3]: In deploying Random IP mutation techniques, this refers to the required address space based on mutation speed (e.g., low-frequency mutation, LFM, or high-frequency mutation, HFM).
  - **Flow table size** [74, 73, 168]: This measures the size of flow table in OpenFlow (OF) switches when OF-RHM (Random Host Mutation) is used in an SDN-based MTD.
  - **Integrated performance cost** [32, 52]: This metric integrates both performance and security cost. Ge et al. [52] considered bandwidth cost and risk at servers upon being attacked to calculate the overall performance cost. Clark et al. [32] defined the cost function in terms of the number of active sessions, caused by triggering MTD operations, and the fraction of decoy nodes scanned by attackers. In these works, the goal is to minimize the performance cost that reflects defense cost, security, and service availability to users.
  - **Strategy switching cost** [135]: This cost measures the switching cost (e.g., migration cost). Sengupta et al. [135] estimated the switching cost in switching web-stack configurations as a cost metric.

– **Power consumption** [167]: When MTD is deployed in resource-constrained environments such as wireless sensor networks or IoT environments, energy consumption is one of the key design considerations. How much benefit is introduced over the power consumption by deploying an MTD technique is a critical metric to measure the efficiency of the MTD.

The metrics to measure the efficiency of MTD by a defender's perspective are summarized in Fig. 2.10 based on 25 papers published during 2011-2018. Since more than one metric can be used in a paper, the number of metrics countered is not the same as the total number of works examined in this literature review.

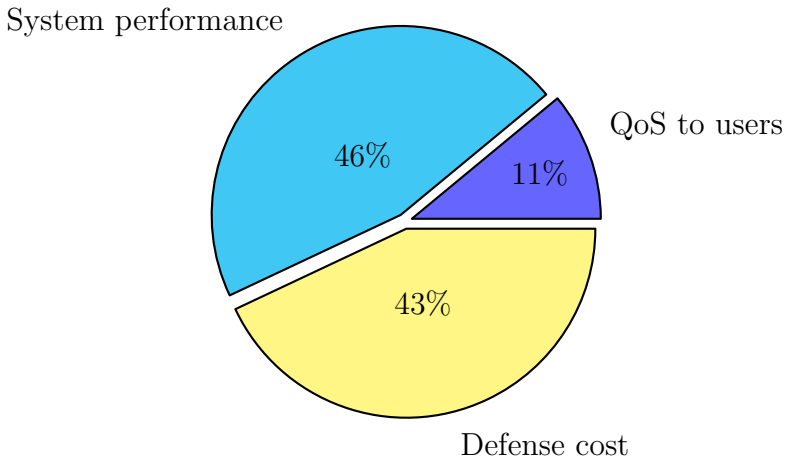


Figure 2.10: Metrics measuring the efficiency of the MTDs by a defender's perspective.

2.4.3 Discussion on Existing MTD Metrics

Different types of metrics have been proposed for assessing the effectiveness and efficiency of the MTDs. The majority of the existing metrics have been designed based on attacker's and defender's perspectives. The general trends observed from this study are: (1) the attack success probability (e.g., whether an attacker achieved its goal of a launched attack such as finding a vulnerable

target) is a dominant metric used for the effectiveness of MTD aiming to minimize this metric; (2) due to a large volume of game-theoretic approaches in the state-of-the-art MTD techniques, the payoff or utility of attackers or defenders is also one of dominant metrics used in the existing MTD techniques; and (3) some system-level metrics measuring system vulnerability or reliability (e.g., degree of vulnerability, MTTF or mean time to compromise an entire system) in the presence of attacks are observed as major metrics used to measure MTD effectiveness. Similarly, as demonstrated in Fig. 2.10, most metrics measuring MTD efficiency belong to system performance or defense cost. However, the level of QoS provided to users is significantly less studied. This means MTD technology focuses more on enhancing system security and performance with minimum cost while service available for users to provide seamlessly, continuous service provision has remained much less explored in designing MTD techniques. Since deploying and executing MTD mechanisms to a system introduces a critical tradeoff between security, defense cost, and service availability, an optimization problem with these conflicting dual goals should be investigated in-depth to meet multiple criteria from both system goals (i.e., multi-objective optimization problem) [28].

2.5 Summary

MTD takes a non-conventional perspective toward defensive security, which aims to enhance security by changing the attack surface, rather than eliminating all the vulnerabilities of system components, representing the conventional security goal. Based on this principle, MTD can provide a way to build proactive, adaptive, and affordable defense mechanisms that can leverage the existing technologies (e.g., SDN or Cloud) while enhancing system security. A large volume of the network address (e.g., IP, port) shuffling MTD mechanisms have been proposed for the different application domains (e.g., conventional network, SDN, Cloud). However, there are limited IP

and/or port shuffling MTD mechanisms that have been proposed for the SDNs. Besides, they often deploy in a conventional network and lack leveraging SDN technology. Leveraging the SDN technology enables flexibility to implement the operations of the MTD mechanisms. Therefore, there is a need for the development of SDN-based MTD approaches to effectively shuffle IP address and/or port numbers of the host of a target system.

The effectiveness and efficiency of MTD are measured by using a variety of metrics in terms of the perspectives of both an attacker and a defender. Attack success probability, attack utility, mean time to compromise, system security (e.g., confidentiality, integrity, and availability), and defense cost, system performance are the popular metrics used to assess the effectiveness and efficiency of the deploying MTD techniques respectively. However, the metrics have not yet well discussed on how to capture the dynamics of the network and/or system properties introduced by the MTD and measure their effectiveness.

Chapter 3

Flexible Random Virtual IP Multiplexing

This chapter presents the proposed MTD mechanism, named FRVM. The FRVM is an SDN-based MTD technique that provides a flexible network address shuffling mechanism aiming to thwart network reconnaissance and scanning attacks. We introduce FRVM defense mechanism in Section 3.1, consider system and attacker models are described in Section 3.2 and Section 3.3 respectively. The detailed explanation of our proposed MTD approach (e.g., FRVM), including system architecture, algorithms, and communication protocols is presented in Section 3.4. The implementation and security evaluation of the proposed FRVM is discussed in Section 3.5. Similarly, the performance overhead of the FRVM deploying in SDN is analyzed in Section 3.6. Section 3.7 summaries the work presented in this chapter.

The main contributions of this chapter are summarized as follows:

- Propose a novel SDN-based MTD technique that provides flexibility to have multiple, random, time-variant IP addresses in a host, and this creates an environment with high diversity that significantly decreases the attacker success rate;

- Develop system architecture, algorithms, and communication protocols

of the FRVM which is feasible to deploy on an SDN-based network infrastructure;

- Derive probabilistic analytical models for analyzing and evaluating the effectiveness of the FRVM;
- Analyze the performance of the FRVM in terms of attacker success probability under scanning attacks and defense costs considering the end-to-end delay and throughput overheads.

3.1 Introduction

The concept of MTD has been introduced to increase uncertainty and confusion for attackers by continuously and dynamically changing the attack surface on a system. The dynamic defense based on MTD techniques has been often applied at different levels of the system [119]. At the system level, the example MTD techniques aim to change the system and/or system's properties dynamically in terms of randomizing instruction sets [16, 83, 127], run-time environment [163], address space layout [137], VM migration [122], OS-rotation [151]. At the network level, the common MTD techniques are related to the shuffling of the network attributes, including mutation of IP address [3, 10, 74], port hopping [93, 143], route mutation [44], and/or network reconfiguration [65]. The well-known existing network address shuffling MTD techniques [3, 74, 73] randomly mutate an IP address of a host mapping to a virtual IP address in a one-to-one manner. One-to-one mapping requires more virtual IP addresses to satisfy the mutation rate constraint and unpredictability for the hosts in the network, which often results in a lack of scalability due to a limited address space. Also, they have been evaluated for the effect of scanning worm attacks, which cannot determine the exact attacker's work effort in terms of scan success based on the scanning time; and lack to investigate the performance overhead incurred due to the deployment of the MTD mechanism

on the SDN environment.

In this chapter, we propose a novel MTD technique called *Flexible Random Virtual IP Multiplexing* (FRVM) in an SDN-based environment that enables a host to have multiple, random, and/or time-varying virtual IP addresses (*vIPs*). FRVM creates short-lived, random *vIP* address(es) for each service of the host. These *vIP* addresses are mapped to a real IP (*rIP*) address of the host. It allows a full range of *vIPs* to the host for different services by mapping in a $M-to-N$ manner where M refers to the number of *rIP* and N is the number of *vIP*. The *vIP* addresses are changed randomly and frequently with a certain time interval in order to invalidate the target system's information collected by the attacker and lower down predictability of the changing patterns of the addresses. The *rIP* address of the host remains unchanged and transparent to the end-user. This SDN-based MTD solution defends against network reconnaissance and scanning attacks. We consider scanning attacks that are typically used for gathering the system's security-related information before launching the actual attacks. The attacker often uses a customized scanning software tools for collecting the target system's information such as operating system types, IP addresses, TCP/UDP open ports, running services, protocols, network topology, and/or exploitable vulnerabilities. The proposed FRVM provides high network diversity in terms of IP addresses of the hosts that can significantly disturb the attacker's scanning strategies and accordingly reduces the attacker's success.

3.2 System Model

Software-defined networking is a promising technology that can provide flexibility, robustness, and programmability [136]. The flexibility and programmability of the SDN technology can be applied to several cybersecurity network applications. The SDN architecture decouples the network control and forwarding functions enabling the network control to become directly

programmable and underline infrastructure to be abstracted for applications and network services [88]. An SDN-based environment consists of SDN-switches (i.e., Open vSwitches) and SDN-controllers (e.g., Ryu [133], OpenDaylight [108], ONTO [19], *etc.*), communicating over a secure channel. The controller on the SDN is responsible for controlling the forwarding operations of the switches in a centralized manner. The SDN controller can enable a switch to rewrite the packet headers and steer the traffic towards a specific host by simply updating the forwarding table of the switch via an OpenFlow (OF)-protocol [107]. Each SDN switch has its flow table with flow rules. The SDN-switch routes packets based on the flow-rules defined in the flow-table. The SDN-switch forwards a packet to the controller if it does not match the flow-rule installed in the flow-table. The controller adds a new flow rule for that packet or drops it. The end-hosts are connected with OF-switch. The programmable interfaces afforded by the SDN-based environment can be comforted to achieve a proactive, dynamic defensive strategy based security including moving target defense.

We consider the SDN-based system model in which end-hosts are called server-hosts, which offer several services to end-users. Each server-host should have at least one active service where a server can have multiple services (e.g., web services, application services *etc.*). TCP/UDP port numbers identify the active services running on the server-hosts, and a *vIP* is assigned to each active service of the server host. A combination of *vIP* and port number (i.e., *vIP: Port*) can be used to communicate within the SDN network infrastructure. Fig. 3.1 describes our system model consisting of the control plane with a controller (e.g., an SDN controller) and the data plan with switches and hosts (e.g., Open vSwitches and server machines).

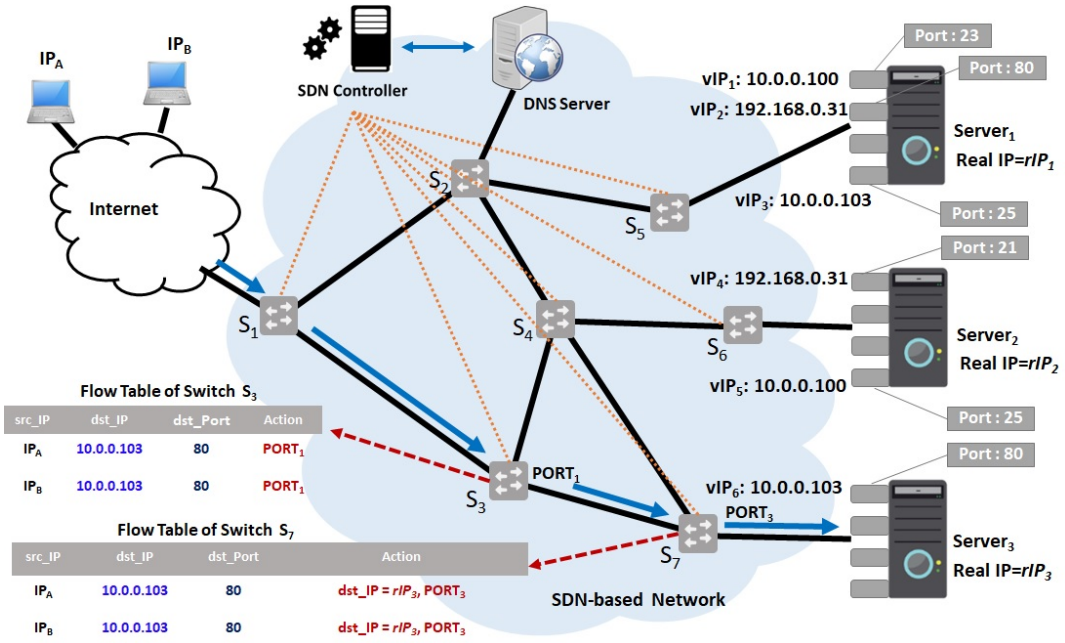


Figure 3.1: FRVM System Architecture

3.3 Attacker Model

Network address shuffling-based MTD mechanism disrupts attackers at the early stage of the attack process. We devise an attacker model based on Cyber Kill Chain (CKC) [70], where the attacker seeks to perform reconnaissance on the target network while the system defense mechanism (e.g., FRVM) aims to hide the IP addresses and/or services. The attacker selects a random IP address and performs network and port scanning to discover the server-hosts (e.g., used IP addresses) and active services (e.g., TCP/UDP ports) for planning and launching the attacks. The attacker adopts a non-repeat random scanning strategy where the attacker scans an IP only once and discards it (i.e., never scans the same IP again). In this work, we assume that the attacker has a certain knowledge of the target system and its defense capabilities. Hence, we make the following assumptions:

- An attacker is located outside the network;
- There are N total IP addresses (i.e., virtual IP address pool space), n number of hosts and m-active currently running services on each host;

- An attacker targets to discover a server-host and its active services;
- The attacker is aware of the virtual address pool size and will sequentially attempt k connections or probes;
- The attacker can use scanning software tools (e.g., *Nmap* [101] and/or *Nessus* [111]) to discover all hosts, services and/or vulnerabilities on the target host;
- The multiplexing event of the FRVM dynamically remaps all *vIPs* at the server-host by uniformly selecting *vIPs* at random and the attacker is aware of the MTD technique running on the target network that shuffles the IP addresses.

3.4 Proposed Approach

In this section, we present the description of each component of our proposed MTD approach FRVM, which includes IP-mapping, system architecture, mapping algorithms and communication protocols.

3.4.1 IP Mapping

In FRVM, each server-host has a real IP address, *rIP*, where the *rIP* is mapped to a set of virtual IP addresses, *vIPs*, and a set of *vIPs* are mapped to an *rIP* in an 1-to-n/n-to-1 manner, respectively. The mapping of one or more *vIPs* to an *rIP* (i.e., *vIPs*-to-*rIP*) of each server-host is called 'virtual IP multiplexing' while the mapping of a *rIP* to *vIPs* mapping is called 'de-multiplexing.' The *vIPs* are unused, private IP addresses (e.g., 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16). We define the multiplexing function in Eq. (3.1) that describes the multiplexing of a set of *vIPs* (e.g.,$vIP_1, vIP_2, \cdots, vIP_m$) to a set of *rIPs* (e.g., a rIP_1) in m-to-1 manner. Similarly, the de-multiplexing function in Eq. (3.2) details the de-multiplexing of a set of *rIPs* (e.g., rIP_1) to a set of *vIPs*

(e.g., $vIP_1, vIP_2, \cdots, vIP_n$) in an 1-to-$n$ manner. To perform multiplexing/de-multiplexing of IP addresses, Eqs. (3.1) and (3.2) combine a *vIP* with a port number (e.g., $vIP : port$).

$$f_{multiplexing} : \mathbb{VIP} \times \mathbb{P} \rightarrow \mathbb{RIP} \tag{3.1}$$

$$f_{de-multiplexing} : \mathbb{RIP} \rightarrow \mathbb{P} \times \mathbb{VIP} \tag{3.2}$$

where $\mathbb{RIP}, \mathbb{VIP}$, and $\mathbb{P}$ denote finite set of all rIP addresses, vIP addresses, and TCP/UDP port numbers respectively. As described in Section 3.3, each server-host offers one or more services, and these services are identified with the corresponding port number. A new *vIP* is generated randomly using a cryptographically secured random number generator and assigned to the server host's service. All the randomly assigned *vIPs* to the server-host are changed continuously based on a short period of the interval, called 'multiplexing interval,' denoted by T. The *rIP* of the server-host remains unchanged while *rIP* is hidden in the network, and routing is restricted to using *vIPs*. The SDN controller handles all the mapping and remapping of addresses, changing of *vIPs* of the server-host, and updating the flow-table entry of SDN switches (e.g., OF-switches). The controller performs mapping of *vIPs*-to-*rIP* or *rIP*-to-*vIP* at edge switch (i.e., just before the end-host). The mapping is transparent to an end-user with no service disruption since *rIPs*, and corresponding port numbers of a server-host remain unchanged. We assume that no one can reach to the server-host using the *rIP* except the authorized network administrators. A user can communicate either using a domain name/host-name or *rIP*, which is described as the communication protocol in FRVM in Section 3.4.3.

3.4.2 System Architecture of FRVM

FRVM defense mechanism is designed to deploy on SDN-based network infrastructure. Fig. 3.1 depicts the FRVM architecture that comprises the major components such as end-hosts, SDN controller, SDN switches, data plane, and control plane. The end-hosts are connected with the SDN-switches in the data plane. The SDN controller centrally controls data forwarding network elements. The components of the FRVM system architecture can be represented by:

- *End-host(s)*: A set of end-hosts includes servers, workstations, and/or gateway interface to routers, denoted by $\mathbb{H}$=$\{Host_1, Host_2, \cdots, Host_n\}$ where $|\mathbb{H}| \geq 2$. Each end-host has an *rIP* assigned to it and a set of *rIPs* is denoted by $\mathbb{RIP}$ =$\{rIP_1, rIP_2, \cdots, rIP_n\}$. A *rIP* is mapped to one or more *vIPs*, and a set of all *vIPs* is denoted by $\mathbb{VIP}$ = $\{vIP_1, vIP_2, \cdots, vIP_m\}$. Each server host offers a number of active services listening on port numbers. A set of port numbers, denoted by $\mathbb{P}$=$\{port_1, port_2, \cdots, port_m\}$.

- *SDN Controller*: A set controllers, denoted by $\mathbb{C}$=$\{C_1, C_2, \cdots, C_r\}$. We assume that a functional SDN network has at least one controller where $|\mathbb{C}| \geq 1$. In our system, a single SDN controller exists that performs the mapping of *rIP*-to-*vIP* or *vIP*-to-*rIP*, installs and updates necessary flows in the SDN-switches. The SDN controller randomly and dynamically changes *vIPs* of end-hosts and *rIPs* of the end-host machines remain unchanged.

- *SDN-switches*: A set of SDN-witches, denoted by $\mathbb{S}$ =$\{S_1$,S_2, $\cdots$, $S_l\}$. Each SDN-switch forwards the data plane traffics based on the flow-rule specified by the SDN controllers. Each switch, S_i, includes ingress and egress ports.

- *Data plane*: Representing a topological connectivity based on a set of

network switches and end-hosts.

- *Control plane*: Representing relations between switches and controllers.
- *End-users*: A set of legitimate users, denoted by $\mathbb{U}=\{u_1, u_2, \cdots, u_j\}$ that access the services running on the servers in $\mathbb{H}$.
- *DNS servers*: The domain name system (DNS) server resolves a domain name of the server-host and returns a *rIP* address.

The IP multiplexing algorithm for an SDN-controller is presented in Algorithm 1. The SDN-switch sends every unmatched packet to the controller which determines the types of communication (e.g., domain-name or *rIP*), performs the translation of IP addresses, and installs necessary flows. The SDN-controller maps a real IP address of the destination host to a number of virtual IP addresses and then updates the flow table of each SDN-switch (e.g., OF-switch). For example, Table 3.1 shows the state of a flow-table of an OF-switch where destination *rIPs* are replaced with the *vIPs*. Table 3.2 shows the mapping of 1-to-n/n-to-1 IP addresses using this algorithm. All *rIPs* hidden within the SDN network packets are forwarded using the *vIPs* only. The SDN controller dynamically changes *vIPs* for each service of a server-host in every multiplexing time interval, T.

Table 3.1: Flow-table information of an SDN-Switch.

src_IP	dst_IP	dst_Port#	Action
IP_A	10.0.0.103	80	Forward to $PORT_1$ with dst_IP 10.0.0.103 and dst_Port# 80
IP_B	10.0.0.103	80	Forward to $PORT_1$ with dst_IP 10.0.0.103 and dst_Port# 80
IP_C	10.0.0.100	23	Forward to $PORT_2$ with dst_IP 10.0.0.100 and dst_Port# 23
IP_D	192.168.0.31	21	Forward to $PORT_3$ with dst_IP 192.168.0.31 and dst_Port# 21

3.4.3 Communication Protocol

We design a typical communication protocol for the FRVM defense mechanism. An end-user of the system deploying the FRVM can have two ways of communicating either using a 'domain-name/host-name' or using a

Algorithm 1: Algorithm for Multiplexing IP Addresses

```
for all packets p from host h_i to h_j do
    if p is a Type-A DNS response for host h_j then
        Set dstAddr(p):=vIP(h_j):dstPort;
    else
        if h_i is authorized access to h_j then
            Set dstAddr(p):=vIP(h_j):dstPort;
        end
    end
    if p is at source SDN-Switch then
        Set scrAddr(p):=vIP(h_i):srcPort;
        Set dstAddr(p):=rIP(h_i):dstPort;
    end
    if p is at destination SDN-Switch then
        Set dstAddr(p):=rIP(h_j):dstPort;
        Set scrAddr(p):=vIP(h_j):srcPort;
    end
    for each service s of host h_i do
        Select a new vIP for each service s of h_i ;
        Update flow table entry;
    end
end
```

rIP address. We assume that a normal user communicates using the domain name, and only an authorized user (e.g., a network administrator) is allowed to communicate using the *rIP* of the destination host, which is shown in Figs. 3.2 and 3.3, respectively. Fig. 3.2 shows that when a DNS query is sent to resolve the domain-name (e.g, *d_name*) of an end-host (e.g., $Host_2$), then the DNS response is intercepted by the SDN controller and the *rIP* of the destination host (e.g, rIP_2) is replaced with the virtual IP address (e.g., vIP_1). As a result,

Table 3.2: IP address multiplexing/de-multiplexing examples.

Real IP address	TCP Port #	virtual IP address
rIP_1	23	10.0.0.100
	80	192.168.0.31
	25	10.0.0.103
rIP_2	21	192.168.0.31
	25	10.0.0.100
rIP_3	80	10.0.0.103

the source end-host (e.g., $Host_1$) receives only a *vIP* (e.g., vIP_2) mapping to the destination host and initiates its communication with the destination service port number (e.g., $Port_2$). Similarly, a source host's *rIP* (e.g., rIP_1) of an outgoing packet is replaced by a randomly selected *vIP* (e.g., vIP_1). The SDN controller installs necessary flows in the SDN-switches in the route. Further, packets will be matched and forwarded by the SDN-switches according to the installed flows in the flow table. The translation of the *vIP*-to-*rIP* will be applied at the SDN-switch before the destination end-host. Fig. 3.3 shows how an authorized user at a host (e.g., $Host_1$) can reach an end-host (e,g., $Host_2$) using the real IP address. In this scenario, the client initiates connection using *rIP* (e.g., rIP_2) and port number (e.g., $Port_2$) of the destination host (e.g., $Host_2$), and the SDN controller requesting authorization for a received packet from the source host (e.g., $Host_1$). If access is granted, the SDN controller replaces the *rIP* of the destination host (e.g., rIP_2) with a randomly generated *vIP* (e.g., vIP_2), and then sends the packet to the SDN controller which installs necessary flows in the SDN-switches. The authorization is assumed to be performed per TCP or UDP connection. In the FRVM network, routing is restricted to use only *vIPs* while *rIPs* are hidden within the network. The multiplexing and de-multiplexing functions enable the association between the destination port and the IP address.

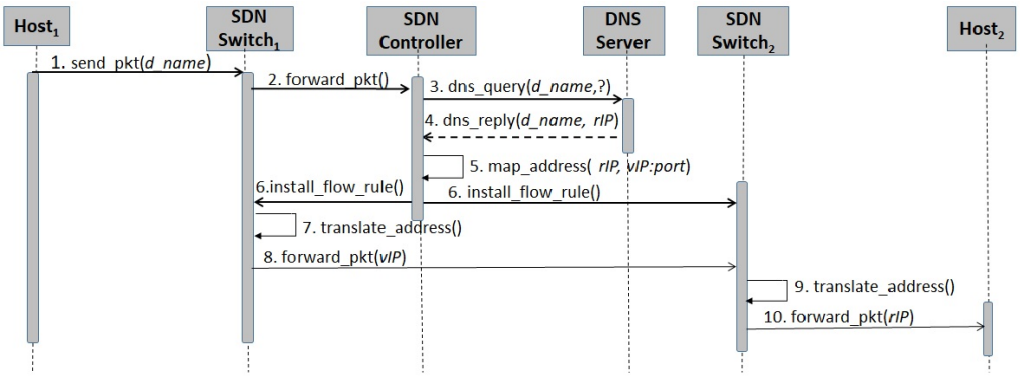


Figure 3.2: Communication via domain-name.

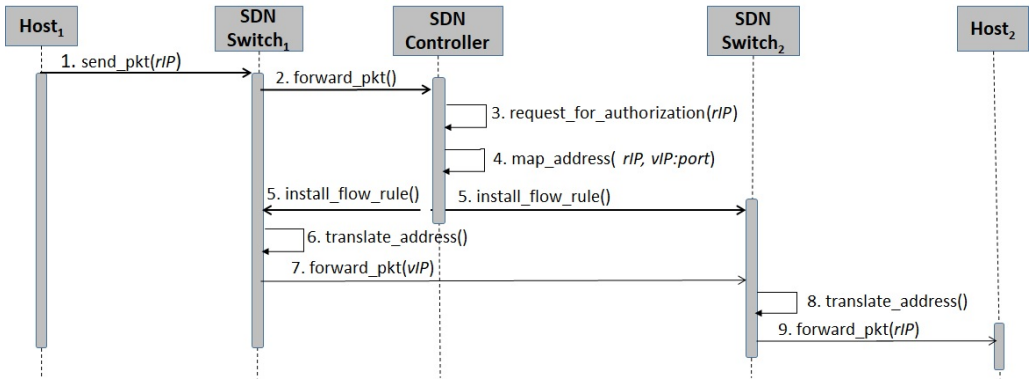


Figure 3.3: Communication via real IP address.

3.5 Implementation & Evaluation

In this section, we present implementation and security evaluation of the proposed FRVM defense mechanism to validate its outperformance over existing counterparts. The validation is performed using multiple methods, including analytical models (e.g., deriving probabilistic models) and simulations with security and performance metrics.

3.5.1 Implementation

We implemented a proof-of-concept FRVM on the SDN-based virtualized networked system. A virtualized network environment is created using Oracle VM VirtualBox [120] and Mininet [91] with Ryu [133] framework. We developed a customized SDN controller called em FRVM controller based on multiplexing algorithm outlined in Algorithm 1 in Section 3.4.2. Routers in the FRVM deploying network are not allocated *vIP* addresses where they contact through their *rIP* addresses, and the protection of the FRVM is not afforded to the routers. Although as routers stitch networks together, they would be reachable through an interface on the connected non-protected network. Therefore, the loss of protection is minimal. Without a *vIP* address, the routers have a static IP address allowing hosts within the network to use them as default gateways. ARP translates MAC addresses to IP addresses within a

network. This functionality is necessary for end-to-end communication. Ideally, ARP packets would be treated similarly to IP packets, multiplexing and de-multiplexing between *vIP* and *rIP* addresses, Unfortunately, multiplexing relies on the inclusion of port numbers, of which ARP packets have none. The solution involved implementing a queue for each host to identify which *vIP* address of a host, an ARP response should be translated to a network edge. This solution could also be applied to other portless protocols like the ICMP. Our implementation proved that the FRVM is feasible for deploying in software-defined networks.

3.5.2 Derivation of Attacker Success Probability

In this section, we derive attacker's success probability distribution functions for comparative study of the performance evaluation of a dynamic network with MTD mechanism and a typical static IP configured network without any other defense mechanisms (i.e., as a baseline model) adapting the existing probabilistic approaches [80] [23] [37]. The notations N, n, k, and x used in the derivation denote the address space size, number of target hosts, number of scans, and number of successes, respectively.

A static network without MTD mechanism: In a static IP configured network, the IP address assigned to an end-host remains unchanged. An attacker's strategy for discovering the end-host is to scan sequentially and iterate through the address space. The attacker scans IP addresses adopting a *non-repeat* random scanning strategy where he never repeats the address which has been scanned before. We can model this scanning strategy with a hyper-geometric distribution as the number of successes in a sequence of scans from a finite set of IP addresses (i.e., address space) without replacement. Therefore, the attacker success probability to discover the host(s) is determined using the hyper-geometric distribution. Accordingly, the probability that the attacker successfully obtains exactly x of n hosts in the address space N in k scans can

be given by:

$$P(X = x) = \frac{\binom{n}{x}\binom{N-n}{k-x}}{\binom{N}{k}} \tag{3.3}$$

where $k \geq x$, $N \geq n$, and $n \geq x$. The probability that the attacker successfully discovers at least one target host is:

$$\begin{aligned} P(X \geq 1) &= 1 - P(X = 0) \\ &= 1 - \frac{\binom{N-n}{k}}{\binom{N}{k}} \end{aligned} \tag{3.4}$$

A dynamic network with MTD mechanism: In our work, the multiplexing or de-multiplexing event (i.e., shuffling event) remaps all *vIPs* to the end-hosts. We assume that the FRVM remaps all the IP addresses of the end-hosts in every scanning attempt. An attacker's success probability remains the same for every scanning attempt. The attacker loses any reconnaissance knowledge gained in every multiplexing event (i.e., every T interval time). The attacker's scan success probabilities are equal in each scan. We model this scenario with a binomial probability distribution, and the attacker's success is determined as the number of success in a sequence of k-scans from a finite set of IP addresses (i.e., address space) with replacement. The binomial probability distribution function with the success probability p is:

$$P(X = x) = \binom{k}{x} p^x (1-p)^{k-x} \tag{3.5}$$

where $p = \frac{n}{N}$ is the probability that the attacker discovers a host in each scan and $k \geq x$. The probability that the attacker successfully discovers at least one target host in k scans is given by:

$$\begin{aligned} P(X \geq 1) &= 1 - P(X = 0) \\ &= 1 - (1-p)^k \end{aligned} \tag{3.6}$$

If the attacker scans whole address space (i.e., $k = N$) with $p = \frac{1}{N}$ (i.e., single target) then Eq. (3.6) reduces to

$$P(X \geq 1) = 1 - (1 - \frac{1}{N})^N. \tag{3.7}$$

As the network address size N increases to a sufficiently large number (i.e., $N \to \infty$), the success probability converges to $1 - (1 - \frac{1}{N})^N = 1 - e^{-1} \approx 0.63$.

3.5.3 Simulations & Experimental Setup

In this work, we focus on evaluating the effectiveness of the FRVM under scanning attacks with respect to the attack success probability. As discussed in Section 3.3, the attacker seeks to perform reconnaissance on the network while the defense mechanism aims to keep the IP addresses (e.g., *vIPs*) hidden. An attacker selects a random IP address to discover a target server-host, and then performs port scanning on the host to discover the active services. We consider a random scanning strategy (e.g., non-repeat scanning) to discover IP addresses of the target server hosts and active service running ports of the target server host. The host discovery (e.g., IP addresses) can be achieved by sending an ICMP message to the target server host. Similarly, discovering active services can be achieved by sending an SYN packet to the target server using port scanning. We used security and performance metrics to evaluate the effectiveness of the FRVM and its counterpart, a static network that uses a static IP configuration. The simulation experimental environment (e.g., a virtualized SDN testbed) is created with a Mininet v2.2.2 [91] emulator using Oracle VM VirtualBox 5.2.12 [120] Hypervisor, Linux Ubuntu 17.10 [96] operating system, Intel i7-3770K with 4.20GHz CPU, Ryu v4.25 [133] SDN controller and OpenFlow switch v2.8.1. We performed our experiments on the virtualized testbed network with the FRVM controller, as described in Section 3.5.1 and without FRVM (i.e., a typical static SDN controller).

We performed TCP connect port scanning using *Nmap* [101], and the

attacker's scan success is computed based on the level of information disclosure (i.e., host and ports discovery) and scan duration. The duration of each scan is collected as it directly impacts the amount of information collected by the attacker that remained valid at the end of the scan. Before the attacker scanned, each protected host randomly selected 10-ports to open in the inclusive range of 50,000-50,999. This range is assumed to contain the 1000 most likely open ports and was consolidated into a single block for simplicity. The port numbers of each service are generated with a cryptographically secure random number generator. The attacker scanned the port range 50,000-50,999 (e.g., 1000 most likely open ports) over class C IP addresses of IPv4. It is assumed that the attacker knew the size of the network and the port range of randomly opened ports. The network size could be identified using utilities *Whois* [40]. The port range was known from a list of well-known TCP ports scanned using Nmap [101].

3.5.4 Simulation Results & Analysis

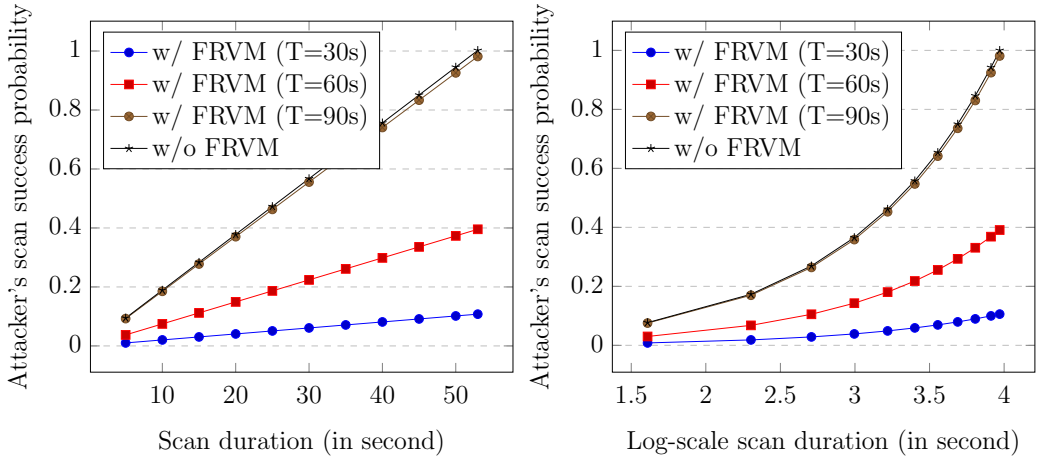
The scan duration was collected from network scans of the virtualized network with FRVM (i.e., w/ FRVM) defense mechanism and without FRVM defense mechanism (i.e., w/o FRVM). Table 3.3 presents the summary statistics of scanning results for the network with FRVM and without FRVM using TCP connect scan under different MTD interval time (i.e., $T = 30s$, $60s$, and $90s$). For each variable a median is listed with lower and upper using the bootstrap method [46]. The results shows that FRVM significantly increases the scan duration consequence the attacker's effort increases to discover the host and services. We analyzed the results with some security metrics.

Attacker's scan success probability: We use scan time-based attacker's success probability metric to evaluate the FRVM for the data sets obtained from the testbed experiments presented in Table 3.3. Connell et al. [35] modeled the MTD's performance defining the attacker's success probability based on a function of time. We comparatively analyzed the effectiveness of the FRVM

Table 3.3: Summary statistics of scanning results.

Network	Scan Duration (in second)			
	MTD interval	$\widetilde{x}$	Lower	Upper
w/ FRVM	30	492.40	279.70	862
	60	133.90	49.91	200.9
	90	54.52	40.3	117.6
w/o FRVM	-	52.89	50.12	54.08

controller against a static typical SDN controller adapting the attacker's success probability with linear and exponential functions [35]. A *linear function*



(a) A linear function (b) An exponential function

Figure 3.4: Comparative analysis of the scanning results in term of attacker's scan success probability for a network w/ FRVM controller and w/o FRVM.

represents the probability of scan success increases linearly with time t and success at time T_s. The probability of attack success in scanning t unit time with the linear function of time can be defined by:

$$P(0 \leq t \leq T_s) = \frac{t}{T_s} \tag{3.8}$$

An *exponential time function* represents attacker's learning in scanning where the attacker initially accumulates knowledge in a low rate, increases scan rate exponentially and becomes more knowledgeable over time, and succeeds at time T_s. The probability of attack success in scanning given t time unit with

an exponential function of time can be defined by:

$$P(0 \leq t \leq T_s) = 1 - \frac{1 - e^{t - T_s}}{1 - e^{-T_s}} \tag{3.9}$$

The Eqs. (3.8) and (3.9) can be used to analyze the attack success with port scanning using the observed total scanning time for a network with FRVM controller (i.e., w/ FRVM) and a typical static SDN controller (i.e., w/o FRVM) respectively. Fig. 3.4 (a) and (b) show attacker's scan success probability for a network with a typical static SDN controller and the FRVM controller using different multiplexing (i.e., MTD) interval time (i.e., T=30s, 60s, 90s) with respect to scan time using Eq. (3.8) and (3.9) respectively. Fig. 3.4a shows that attacker's success slightly increases (i.e., very low rate) with scan duration using the shorter MTD interval time (i.e., T=30s and 60s). However, it grows linearly with scan duration for the FRVM with $T > 90s$ and a typical static SDN controller. Similarly, Fig. 3.4b shows attacker success probability using the exponential function of time. Initially, it has a very small increment with the scan time and becomes exponential. However, the growth rate of the success for the network with FRVM controller is significantly lower than the counterpart SDN controller for $T \leq 90s$. Based on these results, our FRVM effectively increases the work effort of the attacker in terms of scanning time for discovering all the hosts and active services (i.e., network reconnaissance) in the FRVM protected network by 90% with multiplexing interval time $T = 30s$. Moreover, it can be further enhanced its effectiveness by decreasing the multiplexing interval time T. However, there is no significant impact of it with the multiplexing interval time $T \geq 90s$ in our experimental setup described in Section 3.5.3.

Deterrence: It measures the cost exerted on attacker in terms of time by comparing completion time of an attack in the network with MTD and the static network [76]. This metric can be used to evaluate the FRVM using the

results presented in Table 3.3. Let Π denotes the *deterrence* and given by:

$$\Pi = \frac{T_{mtd}}{T_{static}} \tag{3.10}$$

where T_{mtd} and T_{static} denote scan duration for a network w/ FRVM and w/o FRVM respectively. Using the scan duration presented in Table 3.3, we obtained Π=9.30, 2.53, and, 1.03 for the multiplexing intervals $T = 30s$, $60s$ and $90s$ respectively. This shows that FRVM effectively deters the attacker with the lower multiplexing interval time $T \geq 90s$.

Effectiveness: In a network reconnaissance attack, the effectiveness of the deploying MTD mechanism depends on how it increases the attacker's work effort or how it disrupts the attacker. We introduce a metric called *effectiveness* that measures how much the attacker's effort increased by deploying MTD in terms of scan duration. It is denoted by Γ and defined as:

$$\Gamma = \frac{T_{mtd} - T_{static}}{T_{static}} \times 100\% \tag{3.11}$$

Now, computing Γ for different multiplexing or MTD intervals $T = 30s$, $60s$ and $90s$ with scan duration presented in Table 3.3 are Γ=830.98%, 153.16%, and, 3.08% respectively. It shows that the FRVM increases the work effort to the attacker by 830.98% deploying the FRVM with $T = 30s$.

3.5.5 Numerical Results & Sensitivity Analysis

The attacker's success probability is affected by the various environmental parameters such as available address space size, number of scans, number of vulnerable hosts (i.e., target hosts) exist in the network. We analyzed the sensitivity of the parameters to attacker success, discovering the hosts for both a typical static network (i.e., w/o FRVM) and a dynamic network deploying FRVM controller (i.e.,w/ FRVM) based on the probabilistic models described in Section 3.5.2.

Effect of the number of scans: Fig. 3.5a shows the attacker's success probability in scanning the network w/ FRVM and w/o FRVM controller using the address size 2^{10} with respect to the number of scans. The results show that the ASP increases as the number of scans increases. For the typical static network w/o FRVM, the ASP reaches to 1.0 (i.e., 100% scan success) when the attacker scanned the whole available address space. However, in the network w/ FRVM, the ASP slightly increases as the number of scans increases; the ASP reaches up to 0.63 (i.e., 63%) when the attacker scanned the whole address space. This implies that the attacker missed at least 37% of the hosts when he scanned the whole network addresses. These results are obtained for an equal address space pool used by both the attacker and defender to scanning and multiplexing IP addresses, respectively. However, FRVM updates only the small portion (q) of the address space during a scanning time of the attacker. Therefore, the multiplexing rate of FRVM is higher than the attacker scanning rate. In this scenario, ASP in terms of discovering a target host decreases and becomes very low. For example, if the scanner will be able to scan only 10% of the address space (N) and FRVM multiplexes all the IP addresses in this time, ASP will be 9.5% (i.e., $ASP = 0.095$). This implies that FRVM can effectively protect at least 90.5 % of the hosts/services from discovery. Similarly, if the scanner scans only 5% of the total address space, ASP reduces to 4.8% (i.e., $ASP = 0.048$); and for the 1% scan, ASP further reduces to 0.9% ($\approx$ 1%) (i.e., $ASP = 0.0099$). Similarly, Fig. 3.7a depicts the defense performance of the network deploying FRVM controller in terms of the attacker's success probability discovering the host in scanning all the range of private/unused IPs of the class A (e.g., 10.0.0.0/8), B (e.g., 172.16.0.0/12) and C (e.g., 192.168.0.0/16) of IPv4 with respect to the number of scans. The results show that the attacker's success probability is 0.63 in scanning the whole range of IP addresses of each class. The attacker's effort increases (i.e., the attacker's success decreases) in the order of the address space size. With a large number of scans $2^{24} = 1,67,77,216$, $2^{20} = 10,48,576$, and $2^{16} = 65,536$,

the attacker is able to discover only the 63% of the host (i.e., 37% hosts missed) even when scanning the whole private IP addresses of A, B, and C classes respectively. Therefore, FRVM effectively thwarts the network reconnaissance and scanning attacks using the private IP addresses for virtual IP addresses of the protected networked systems.

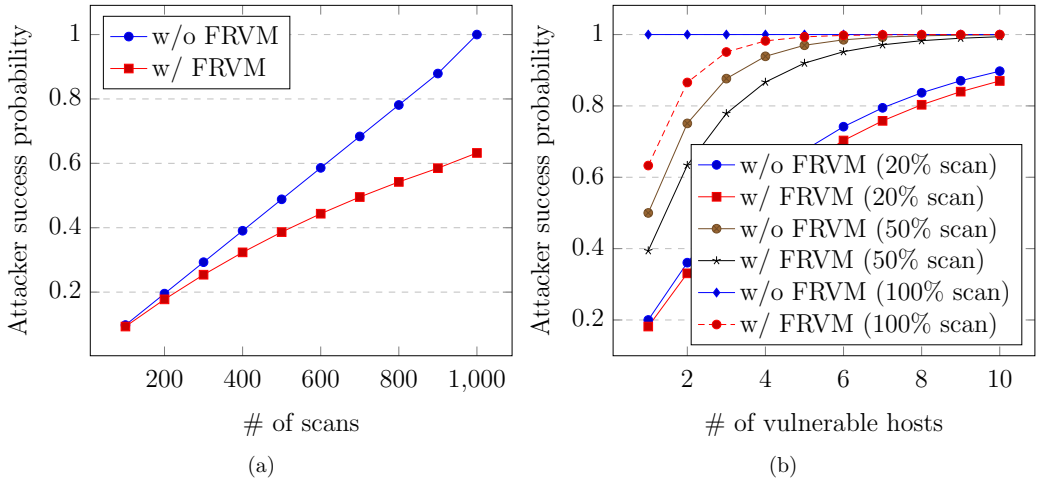


Figure 3.5: Comparative security analysis of the FRVM against a counterpart in terms of ASP to discover the hosts.

Effect of the number of vulnerable hosts: As described in Section 3.5.2, if the number of vulnerable hosts in the network increases, then the attacker's success probability will increase. Fig. 3.5b depicts the attacker's success probability to discover at least one target host in the network containing multiple vulnerable targets with a different percent of the address scanned. The results show that the attacker's success increases as the number of vulnerable hosts and scans increase. The FRVM controller significantly reduces the attacker's success as compared to the static network w/o FRVM controller. Similarly, Fig 3.6a depicts the attacker's success probability to discover the host in the network deploying the FRVM controller varying the percentage of addresses scanned with a different number of vulnerable hosts exist in the network. The results show that the attacker's success probability increases with an increase in the percentage of the address scanned. The effect of the FRVM is more pronounced under a small number of vulnerable hosts in the

network.

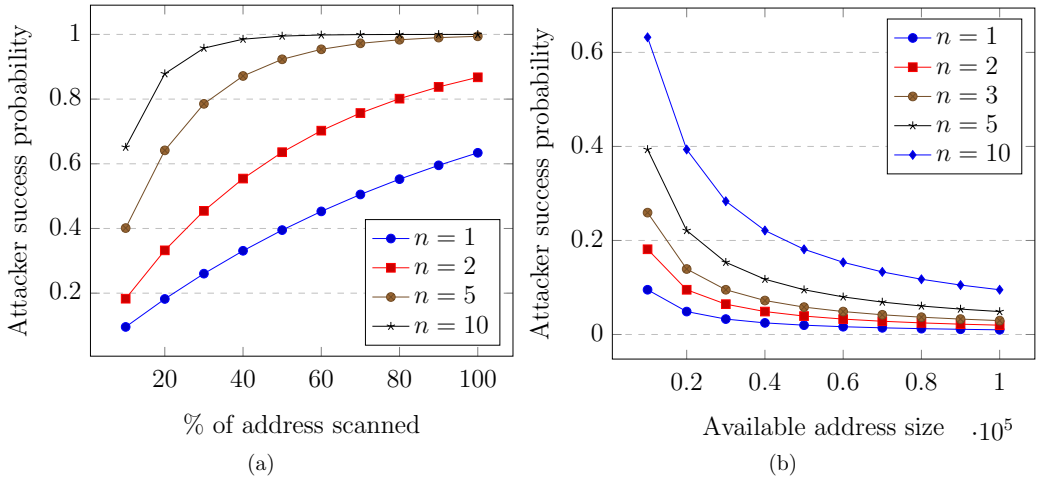


Figure 3.6: Security analysis of the FRVM approach in terms of ASP to discover a host in a network protected with FRVM mechanism.

Effect of available address size: The size of the available address pool has a significant impact on reducing the attacker's success. Fig 3.6b shows the attacker's success probability to discover a target host in a network deploying FRVM with different vulnerable hosts. The results show that the attacker's success dramatically decreased as increasing the address size. However, there is a negative impact on the vulnerable hosts that exist in the network.

ASP to discover multiple hosts: Fig. 3.7b shows ASP discovering the multiple hosts in the targeted network with varying number of successes. The result shows that the attacker's success steeply reduces over the number of successes, and its value is very low for the network w/ FRVM defense mechanism (e.g., FRVM controller) as compared to the static network w/o FRVM mechanism. In this experiment, we set up a small network with 5 server hosts ($n = 5$) and 50 IP address size ($N = 50$) and computed ASP to discover 1 host, 2 hosts, 3 hosts, 4 hosts and finally all 5 hosts for the both networks (e.g., w/ FRVM and w/o FRVM) in 10 scans or probes (e.g., $k = 10$. The result shows that FRVM gradually reduces ASP to successfully discover multiple targets as compared to the typical static network without the FRVM mechanism.

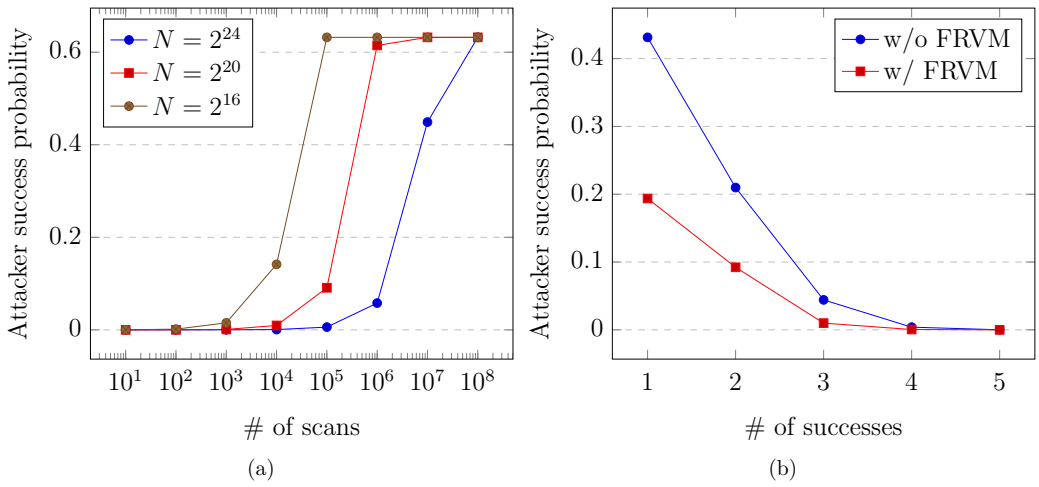


Figure 3.7: Attacker success probability to discover the vulnerable host(s): (a) using FRVM for the range of private IP addresses of class A, B, and C; and (b) multiple targets in a network w/ FRVM and w/o FRVM.

3.6 Overhead Analysis

In this section, an investigation on the performance overheads (e.g., end-to-end delay, throughput) of the deploying MTD mechanism (e.g., FRVM) in SDN-based networked systems is presented. Also, we analyzed the flow management requirements (e.g., flow-table size, update rate) of the FRVM controller.

3.6.1 End-to-End Delay

In FRVM, an SDN controller (i.e., FRVM controller) handles all the incoming packets from the OF switches. In every multiplexing interval, it generates *vIPs*, maps *rIPs-to-vIP*, installs flow rules, DNS queries/updates, and updates the flow-table entries in OF-switches. The FRVM controller makes flow rules updates on the flow table of the SDN switch parallel via *OFPT_FLOW_MOD* operations. However, it increases the end-to-end delay as compared to the static network (i.e., typical SDN w/o FRVM). This additional end-to-end delay incurs due to the deploying MTD (e.g., FRVM) is called *overhead delay*. It is denoted by τ. We introduce a metric called *overhead*

delay percentage in order to know the quantity of overhead incurred due to the deploying MTD mechanism into the network. It is denoted by Δ, and given by:

$$\Delta = \frac{D_{mtd} - D_{static}}{D_{static}} \times 100\% \tag{3.12}$$

where D_{mtd}, and D_{static} denote the end-to-end delay of the network with MTD and without MTD mechanism, respectively. Fig. 3.9a shows the experimental testbed results of end-to-end delay for the network with the FRVM controller (i.e., w/ FRVM) and typical static SDN controller (i.e., w/o FRVM). The results show that the network deploying FRVM adds $0.00134ms$ delay and $\Delta = 8.8\%$. This overhead (Δ) can be reduced by increasing the MTD shuffling (i.e., FRVM's multiplexing) interval time and vice-versa. Therefore, we further investigated the overhead delay for different MTD time intervals with other parameters. Fig. 3.8 shows how the `Packet-in` message flow in

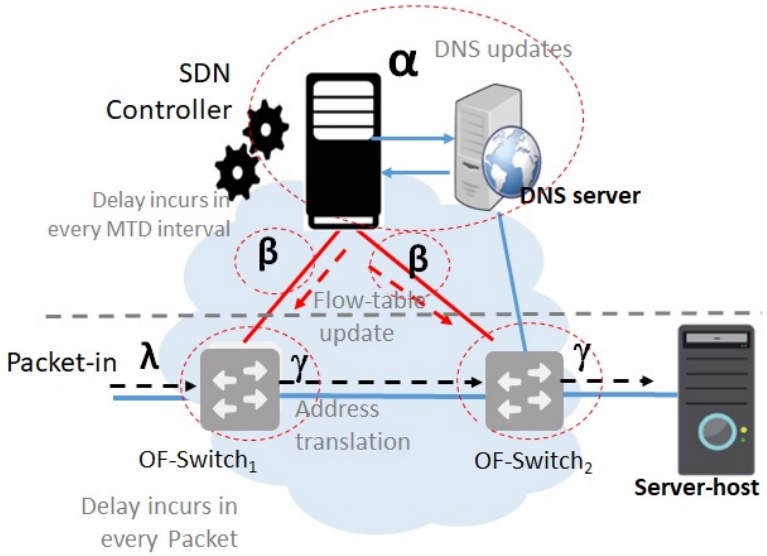


Figure 3.8: Packet-in message flow operations in a network with FRVM controller.

software-defined network with FRVM controller. The FRVM's operations such as DNS queries/updates, virtual address generation and mapping and Flow-table updates cause delay per MTD interval. Similarly, the operations such as address translation (rIP-to-vIP/vIP-to-rIP) and packet header modification

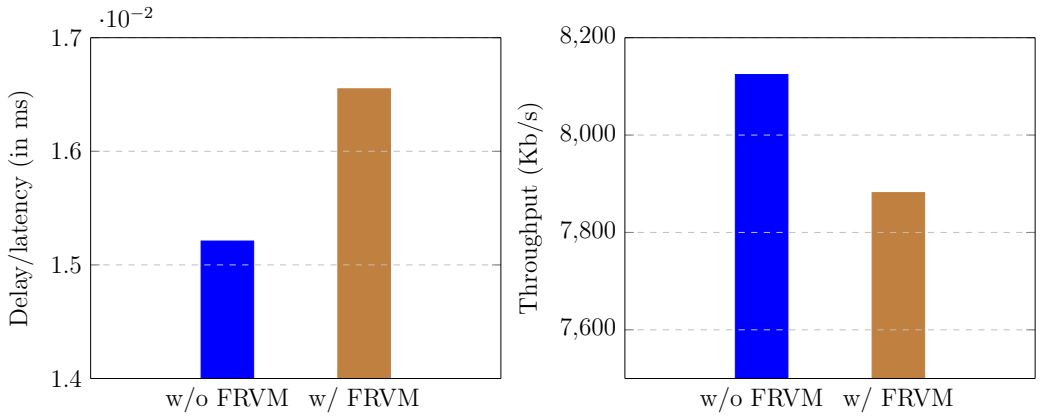
incurs delay in every packet. We define the following parameters for representing the delays incur for the different FRVM's operations.

- α: delay caused for DNS updates and IP address mapping operations at the SDN controller.
- β: delay incurred for updating the flow-table operation at the OpenFlow switch.
- γ: delay caused for address translations (i.e. *,vIP-to-rIP/rIP-to-vIP)* and; and packet modification operations. These operations are performed at the end switches.
- λ: packet arrival per second.

The delay incurred in every MTD interval time with T, and in every packet with λ arrival rate are $\frac{\alpha+\beta}{T}$ and $\frac{2\gamma}{\lambda}$ respectively. The *overhead delay* due to the deploying of the FRVM is a sum of these two delays, and it can be computed using the Eq. 3.13.

$$\tau = \frac{\alpha + \beta}{T} + \frac{2\gamma}{\lambda} \tag{3.13}$$

Fig 3.10 depicts the results of the sensitivity analysis of the overhead delay incurs due to the deployment of the FRVM with varying MTD interval time and packets arrival rate. Fig. 3.10(a) shows the overhead delay of the FRVM varying MTD interval time (i.e., multiplexing interval time) for different packets arrival rate. The results show that the delay substantially reduces as MTD interval time and packet arrival rate increase. Fig 3.10(b) shows that the overhead significantly decreases as increasing packets arrival rate. However, it noticeably increases with decreasing the MTD interval time (i.e., increasing multiplexing frequency). With high multiplexing frequency (i.e., low MTD interval) and low packet arrival rate significantly increase the overhead delay of the FRVM. It can be reduced, reducing multiplexing frequency and increasing packets arrival rate. However, there is a trade-off between security and performance.



(a) End-to-End Delay

(b) Throughput

Figure 3.9: Performance overhead of FRVM controller with a typical static SDN controller.

3.6.2 Throughput

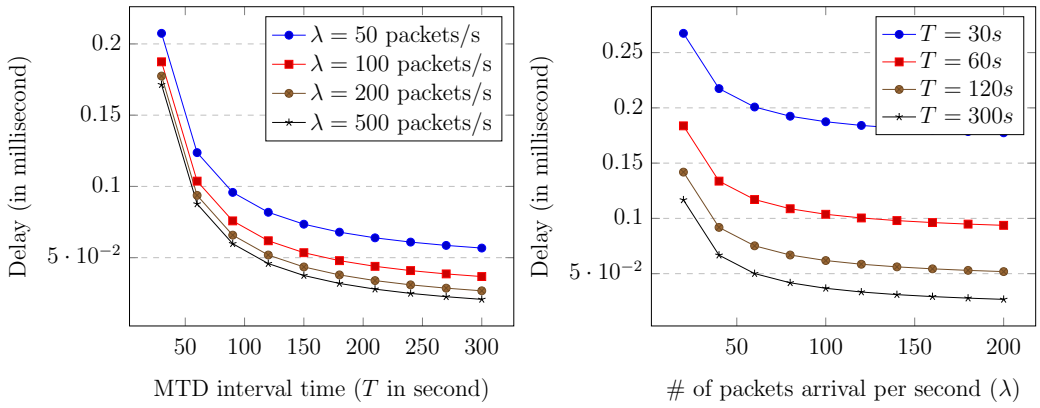
The throughput of the network with FRVM and without FRVM was measured using the duration of file transfers over TCP. The duration of the file transfer was recorded by each of the outside users downloading a 100MB file from protected servers. The file transfer measurements were taken on the virtualized network described in Section 3.5.1. For the transfer, hosts inside of the network acted as servers and outside hosts acted as clients. The bandwidth of simulated links was restricted to $100Mb/s$ and $10Mb/s$ for links within the network and links connecting to the network, respectively. Additionally, the network performance of FRVM controller (i.e., w/ FRVM) is compared with the typical SDN controller (i.e., w/o FRVM).

Let Ω denotes *overhead throughput percentage* of the network deploying MTD (i.e., w/ FRVM controller) with respect to the baseline typical SDN controller (i.e., static network w/o FRVM). It can be defined by:

$$\Omega = \frac{TP_{static} - TP_{mtd}}{TP_{static}} \times 100\% \tag{3.14}$$

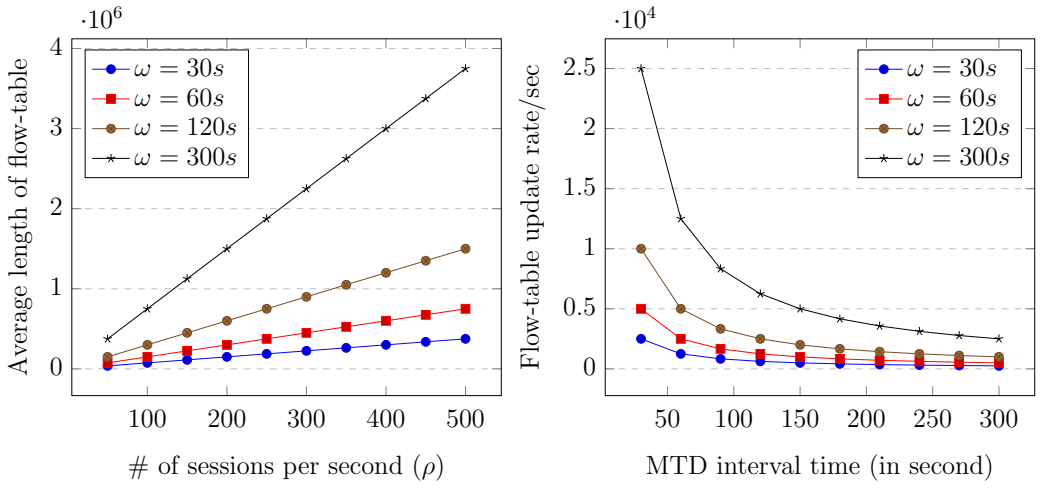
Where TP_{mtd} and TP_{static} represent the throughput of the network w/ FRVM and w/o FRVM respectively. Fig 3.9b shows the throughput measured for the

network w/ FRVM and w/o FRVM. Using these throughput results such as $TP_{mtd} = 7881.773Kb/s$ and $TP_{static} = 8124.302Kb/s$ in Eq.(3.14), we obtained $\Omega = 2.98\%$. This shows that deploying the FRVM controller into the SDN network adds only 2.98% overhead.



(a) Varying T with different λ (b) Varying λ with different T

Figure 3.10: Analysis of performance overhead of deploying the FRVM with $\alpha = 5.0ms, \beta = 0.023ms$, and $\gamma = 1.0ms$.



(a) Varying ρ with different session durations (ω) (b) Varying T for different ω with $\rho = 100$.

Figure 3.11: Analysis of the flow management requirements for deploying FRVM in a network with $n = 5$, and $m = 5$.

3.6.3 Flow Table-Size & Update Rate

Flow table size: In a network deploying FRVM, the flow-rules are specified

for each TCP/UDP session, and two flow entries must be specified for each pair of communicating hosts since the source and destination IP (=vIP) and TCP/UDP ports must be specified. The number of flow-rules in a flow-table of an OpenFlow switch (e.g., Open vSwitch [124]) depend on the number of a host connected with, and the number of active services running on each host. For instance, if a network has n-hosts are connected via an OpenFlow switch and each of the host offering m-active services, the number of flow-entries at the flow-table of the OpenFlow switch are $n \times m$ and the length of the flow-table growth with $O(mn)$. According to the Little's law in queueing theory, if a network has n-hosts which are connected via an OpenFlow switch, each host has on average m-number of active services with ρ-sessions per second, and each session on average takes ω-seconds to terminate, then mean length of flow-table (i.e., average no. of flow-rules) at the OpenFlow switch is denoted by ζ and computed using Eq. (3.15).

$$\zeta = n \times m \times \rho \times \omega \tag{3.15}$$

Fig. 3.11(a) shows the average flow-table length requirements of an OpenFlow switch in a network deploying FRVM controller with 5-hosts, where each host has 5-active services for various session establishments rate and duration. The results show that the length of the flow-table linearly grows as session rate and duration grow.

Flow update rate: In FRVM, the flow-rules are modified and then updated the flow-table in every multiplexing interval time, T. If $\theta = \frac{1}{T}$ denotes an average multiplexing rate per second, then FRVM modifies the flow-rules at θ rate. Let μ denotes an average flow update rate of the FRVM controller, and it can be estimated multiplying an average number of flow-rules, which is mean length of flow-table (i.e., obtained from Eq. (3.15)) by the multiplexing rate. It is given by:

$$\mu = \zeta \times \theta = \frac{n \times m \times \rho \times \omega}{T} \tag{3.16}$$

Fig. 3.11(b) shows the results of the flow update rate requirements for FRVM controller varying MTD interval time (T) with different session duration. The results show that the flow-table update rate significantly decreases an increase of MTD interval time. The length of the session duration negatively impacts the flow-table update rate of the FRVM controller for the high multiplexing frequency and vice-versa. These results show that our FRVM's flow management requirements (i.e.,flow-table size and the flow update rate) are acceptable and feasible with shorter session duration and rate for deploying on SDN-based network infrastructure using the software kernel switches such as Open vSwitch (OVS) [124] which allows 200,000 flow entries [125] with flow-update rate of 42,000 rules per seconds [89].

3.7 Summary

This chapter presented a novel SDN-based proactive and dynamic security defense mechanism as an MTD technique to deal with the network reconnaissance and scanning attacks. The proposed MTD technique is deployed in the SDN-based environment by developing a customized SDN controller called `FRVM controller` and validated its performance against a typical static SDN controller counterpart. The experimental results proved that the FRVM could effectively thwart scanning attacks increasing the attacker's work effort with acceptable operational overhead.

Chapter 4

Random Host and Service Multiplexing

This chapter proposes a proactive SDN-based MTD mechanism called Random Host and Service Multiplexing (RHSM), which obfuscates the real identities of both hosts and their services (e.g., IP addresses and TCP/UDP Ports). RHSM apples the virtual IP multiplexing (i.e., FRVM) concept introduced in Chapter 3 of this thesis to both hosts and services multiplexing of both servers and client systems in a network. The concept of the RHSM security defense mechanism is introduced in Section 4.1. Section 4.2 and Section 4.3 describe the system and attacker models considered in this work respectively. The proposed RHSM approach with system architecture, IP and ports mapping algorithms, and communication protocols are presented in Section 4.4. The simulation experiments and evaluation of the proposed RHSM are discussed in Section 4.5. Section 4.6 summarizes the work presented in this chapter.

The main contributions of this chapter are summarized as follows:

- Propose a novel SDN-based MTD approach that obfuscates real identities of both hosts and services and provides secure communications between the end systems;
- Design an SDN-based architecture, algorithms, and communication

protocols that enable the more effective realization of the proposed RHSM on the testbed;

- Analyze the performance of the RHSM against network reconnaissance and scanning attacks in terms of attacker success probability to discover the hosts and services for the address space size and the vulnerable services;

- Investigate the overhead derived from operating RHSM in terms of the size of the flow table, the frequency of flow rules modifications, and the performance of an SDN controller.

4.1 Introduction

The static nature of conventional networked systems has provided advantages for attackers to efficiently launch attacks and accordingly achieve their attack success while it makes a defender hard to defend against these attacks. Attackers often have asymmetric advantages by taking ample time to monitor a target system, gain the intelligence towards the target system's configurations, identify exploitable vulnerabilities on it, and finally select a time and place to maximize the benefit of launching the attack. To mitigate the high benefits of the attackers, the concept of MTD has been introduced as a proactive defense mechanism to prevent cyberattacks by dynamically changing attack surface to increase an attacker's confusion or uncertainty towards the target systems (or networks) [57]. Hong and Kim [64] categorized the MTD techniques into three classes: shuffling, diversity, and redundancy. Shuffling-based MTD employs rearrangements and/or randomization of system/network configurations in various ways, such as changing instruction set, address space layout, IP address, MAC address, port number, and/or route [3, 10, 74, 73, 79, 99, 103, 139]. The existing techniques using IP address shuffling [85, 3, 10, 76] and port hopping [93, 99, 143] used the random mutation

of IP addresses and port numbers and mapped in a one-to-one manner, respectively. Also, they are often deployed in conventional networks. However, the existing approaches are known to lack of using both the IP address and the port numbers as a single solution for software-defined networking (SDN) environments. The key benefit of using the SDN technology is to separately manage the control panel from forwarding functions, which allows a network to achieve both direct programmability (i.e., high controllability) and abstraction of the underlying infrastructure for the secure deployment of applications and network services [88, 136]. This merit of the SDN can provide a way of efficient and practical designs of dynamic and adaptive MTD techniques compared to other network environments.

In this chapter, we propose *Random Host and Service Multiplexing* (RHSM) as a novel MTD technique to be deployed in an SDN environment to obfuscate the real identities of both hosts and their services (e.g., IP and Ports) during communications. Each server host (i.e., end/server-host) is assigned an IP address, called a real IP (*rIP*) address. A server-host can have one or more active services while each active service has a TCP/UDP port number, called a real port (*rPort*) number. RHSM enables a server-host and service running on it to have random, multiple IP addresses and port numbers that keep changing over time. They are called virtual IP (*vIP*) addresses and virtual port (*vPort*) numbers, respectively. These short-lived *vIP*s and *vPort*s of the server-host are multiplexed (i.e., mapped) to an *rIP* and *rPort* in an N-to-1 manner in multiplexing and a 1-to-N manner in de-multiplexing, respectively. The *vIP*s and *vPort*s of the server-host are frequently and randomly changed to prevent attackers from capturing the patterns of IP addresses being changed by the system. This SDN-based MTD solution is mainly designed to handle scanning attacks (i.e., network reconnaissance).

4.2 System Model

We consider an SDN-based system model to deploy the proposed MTD technique where the SDN consists of end hosts, OF switches, and an SDN controller. The end hosts are further divided into two types: client hosts and server hosts. A *server-host* provides multiple services to client-hosts (i.e., end-users) where at least one service is active. Each service a server-host provides has its ID in terms of the corresponding port number. A *client-host* can access any active services offered by the server using the server host's IP address and port number. We describe our system model in Fig. 4.1 in terms of the control plane with an SDN controller (e.g., OpenDaylight [108]) and the data plane consisting of OF switches and end hosts, including client and server hosts.

4.3 Attacker Model

An attacker is assumed to exist outside the system model. The attacker uses a random scanning strategy and monitors the network (i.e., reconnaissance) while a defense system aims to hide real IDs, such as real IP addresses and port numbers. The attacker tries to discover a target host, scans the target host, exploits its vulnerabilities, and then finally compromise the target host. An attacker is assumed to have a certain level of intelligence and/or information towards a target system and the system's defense mechanisms in place. Hence, we make the following assumptions:

- There are N total IP addresses (i.e., IP address space), m total port addresses (i.e., port address space, $m \leq 2^{16}$) available, and n total server-hosts exist in the network (i.e., $n \leq N$);
- Each of the server-host is assigned to a *rIP* address that has mapped to one or more *vIP* addresses;
- Each server-host has v-active services (i.e., $1 \leq v \leq m$);

- An attacker targets to discover a server-host and its active services;
- The attacker is aware of the address space (i.e., N and m) and will sequentially attempt k probes/scans to find a target host's IP address and ports; and
- The attacker can use a scanning tool (e.g., *Nmap* [101]) to identify both server hosts and services (i.e., ports) actively running on the target server hosts.

4.4 Proposed Approach

In this section, we describe our proposed RHSM approach. The RHSM is an SDN-based proactive MTD technique designed to provide network obfuscation based on the random mapping of *rIP*-to-*vIP*(s) address and *rPort*-to-*vPort*(s) of server-host(s) and vice-versa.

4.4.1 Architecture

The RHSM system architecture consists of four main components: an SDN-controller, OpenFlow (OF) switches, proxies, and end-hosts (e.g., server-hosts & client-hosts). The client-hosts and server-hosts are end hosts while the OF-switches in the data plane connect the end hosts. Fig. 4.1 depicts the system architecture used by our proposed RHSM. Each component of the system architecture is described below.

End-host(s): End hosts include a set of servers, workstations, and gateway interface to routers. Each end-host is assigned with an *rIP* address where the *rIP* is mapped to a short-lived *vIP* address. A server-host can provide multiple active services with *port numbers* (i.e., TCP/UDP ports). The client-host can send a request to access the services provided by the server host.

An SDN Controller: An SDN-controller is the core of an SDN. All the communications between end hosts have to go through the SDN controller,

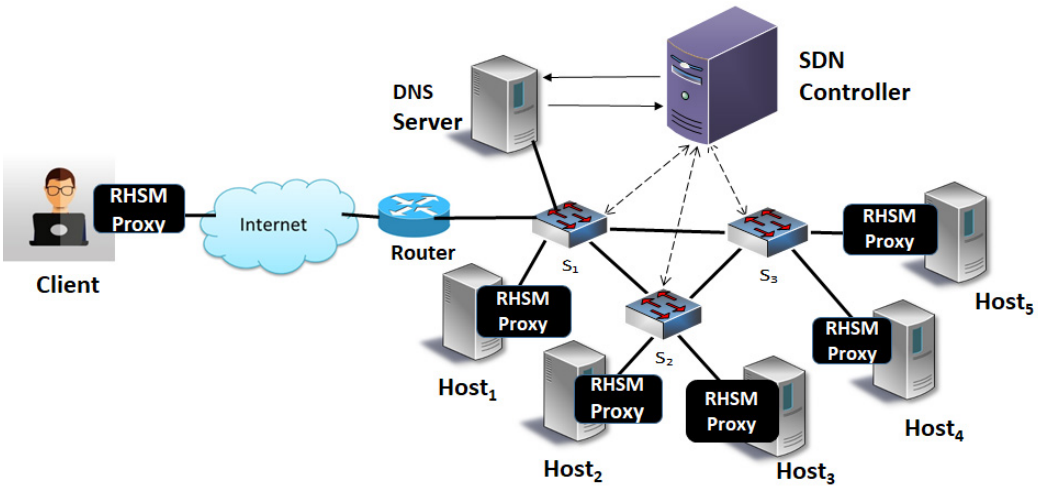


Figure 4.1: System architecture of RHSM.

which installs necessary flows on the OF switches and makes flow table updates. We extend the SDN controller's functionalities to implement the RHSM to provide the multiplexing of IP addresses and ports using the mapping, as in Algorithm 2. We can implement this algorithm by choosing any one of the existing SDN controllers (e.g., Ryu [133],OpenDaylight [108], ONTO [19]). Table 4.2 shows an example demonstrating how the SDN controller conducts the mapping for the *vIPs-to-rIP* and *vPorts-to-rPort*, and vice-versa; and how *vIPs* and *vPorts* are dynamically changed based on a time interval. The SDN controller has the following key roles:

- It generates random *vIP* addresses and *vPort* numbers for the server-host's *rIP* and *rPort*;
- It performs the mapping of *rIP-to-vIPs* of IP addresses and *rPort-to-vPorts* of ports, and assigned them to the servers;
- It dynamically changes the *vIPs* and the *vPorts* based on the corresponding intervals, T_i^{vIP} and T_j^{vPort} respectively; and
- It updates the flow table of OF switches and lookup (mapping) tables of the proxies.

Algorithm 2: RHSM Algorithm for the SDN Controller

```
for all packets p from host h_i to h_j do
    if h_i's access to h_j is authorized then
        if p is a packet to set up a new connection then
            map rIP_{h_j}-to-vIP_{h_j}, and rPort-to-vPort;
            send vIP and vPort to host h_i, and update lookup-table of h_j;
            update the flow table(s);
        end
    else
        drop p;
    end
    for multiplexing intervals, T_i^{vIP} of each host and T_j^{vPort} of each service do
        generate new vIP and vPort;
        update the lookup table(s) in proxies;
        update the flow table(s) of OF-switch(es);
    end
end
```

Table 4.1: An instance of flow table entries of an OF-switch.

MAC src	MAC dst	IP src	IP dst	TCP sport	TCP dport	Action
*	*	IP_1	10.0.0.1	*	9906	$PORT_1$
*	*	IP_2	10.0.0.10	*	7889	$PORT_2$
*	*	IP_3	10.0.0.15	*	8779	$PORT_3$

OF-switches: An SDN consists of a set of interconnected OF-switches (e.g., Open vSwitch [125]). Each OF switch forwards the data plane traffics based on the flow rule specified by the SDN controller. An OF protocol provides an interface for the communications between the OF-switches and the controllers [107]. At every multiplexing interval time, the SDN controller updates the flow rules of the OF-switches using the OF protocol. Table 4.1 depicts the flow table entries of an OF-switch of the SDN system deploying the RHSM defense mechanism.

RHSM proxy: Each end host of the RHSM-adapted network is installed with an RHSM proxy. The RHSM proxy performs the translation and replacement of IP and ports for ingress and egress packets' headers. It performs *vIP-to-rIP* and *vPort-to-rPort* translation for incoming packets. Similarly, it performs the translation of *rIP-to-vIP* and *rPort-to-vPort* for the outgoing

Algorithm 3: Host and Service De/Multiplexing Algorithm for the RHSM Proxies

for *all egress packets p from host* h_i *to host* h_j **do**
  if *p is a packet to setup a new connection* **then**
    randomly generate new vIP for host h_i and new $vPort$ for a service;
    store rIP_{h_i}-to-vIP_{h_i} and $rPort$-to-$vPort$ mapping in lookup table;
    set $p.srcIP := vIP$ and $p.srcPort := vPort$;
  else
    find vIP and $vPort$ of host h_i in the lookup table;
    set $p.srcIP := vIP$ and $p.srcPort := vPort$;
  end
  send secure message(s) to the SDN-controller for vIP and $vPort$ of destination host h_j;
  store rIP_{h_j}-to-vIP_{h_j} and $rPort$-to-$vPort$ mapping in the lookup tables;
  set $p.dstIP := vIP$ and $p.dstPort := vPort$;
  send packet p with vIP and $vPort$ to the OF-switch;
end
for *all ingress packets p from host* h_i *to host* h_j **do**
  multiplex $vIP_{h_j}(s)$-to-rIP_{h_j},$vPort(s)$-to-$rPort$ using the lookup table;
  deliver packet p to rIP_{h_j}: $rPort$ of h_j;
end

Table 4.2: An example of $n - to - 1/1 - to - n$ multiplexing of server-host's virtual IPs and services' virtual ports.

(a) A state at time t_0

rIP	vIP	$vPort$	$rPort$
rIP_1	10.0.0.1	9906	80
	10.0.0.10	7889	
rIP_2	10.0.0.15	8779	1433
	10.0.0.20	6550	3306
	10.0.0.12	5666	
rIP_3	10.0.0.50	9995	443
rIP_4	10.0.0.25	7788	110
rIP_5	10.0.0.10	5550	156
	10.0.0.30	5515	

(b) A state at time t_1

rIP	vIP	$vPort$	$rPort$
rIP_1	10.0.10.10	8908	80
	10.0.10.15	8809	
rIP_2	10.0.10.100	6770	1433
	10.10.0.25	7060	3306
	10.0.10.50	6666	
rIP_3	10.0.10.5	4445	443
rIP_4	10.0.10.3	2210	110
rIP_5	10.0.10.12	1556	156
	10.0.10.65	1510	

packets. This mapping of IP and port of the end hosts is described in Algorithm 3. The RHSM proxies are implemented in both client and server host's end systems of the network.

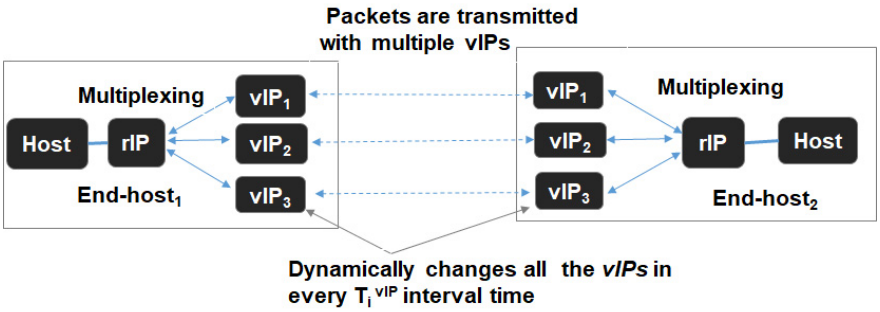
4.4.2 Host Multiplexing

Server-host's vIP multiplexing: This defense mechanism obfuscates the server-host's *rIP* while communicating with replacing the *rIP* with a random short-lived *vIP* in a packet header before transmitting to the network while *vIPs* are changed with a certain time interval, T_i^{vIP}. Fig. 4.2a shows the abstraction of how hosts' *vIPs* are multiplexed and de-multiplexed in the communicating end hosts. The *vIP* addresses for server-hosts are generated using the cryptographically secured random number generator for the address space range of the server-host. The mapping *rIP*-to-*vIP*/*vIP*-to-*rIP* can be performed at proxies on the server-host or client-host. Algorithm 3 describes the translation process of both *vIPs-to-rIP* and *vPorts-to-rPort* and vice-versa, respectively. The mapping is performed transparently to end users without disrupting services since *rIPs* and *rPorts* of a server-host are not changed.

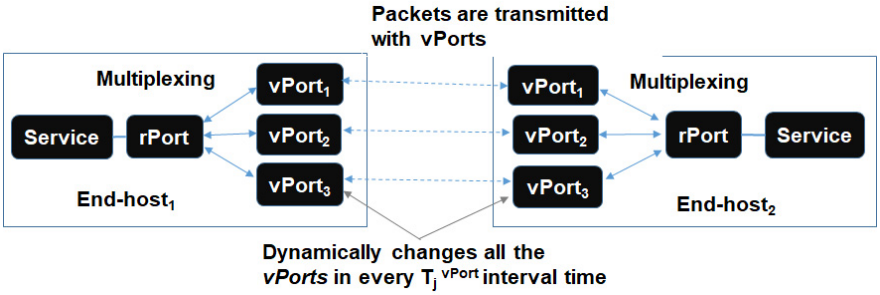
Client-host' *vIPs* multiplexing: The client host's IP multiplexing is performed in order to hide its real ID (*rIP*) in the network and mapping the multiple *vIP* addresses to an *rIP* of the client. The *vIP* replaces the *rIP* of the client and then sends the packet to the network. These clients' *vIPs* are valid per session. The client-host also uses the proxy to perform the translations of real and virtual IPs and ports with rewriting the packet header.

4.4.3 Service Multiplexing

The service multiplexing is a defense mechanism to hide the actual port numbers (i.e., *rPorts*) of the running services of a server-host with de/multiplexing of a service's *rPort* to the multiple *vPorts*. Each *rPort* of a server-host can be mapped to a set of short-lived, random *vPorts*, which are changed constantly based on a certain time interval, T_j^{vPort}. Therefore, an active service running on a server-host can have multiple, virtual port numbers, *vPort*, and the multiplexing operation can be mapped (i.e., multiplexed), such as multiple *vPorts* numbers to a single actual port (i.e., *rPort*) number of



(a) Host Multiplexing: 3-virtual IPs of the host are multiplexed to a real IP.



(b) Service Multiplexing: 3-virtual ports of service are multiplexed to a real port.

Figure 4.2: Host and service multiplexing operations in RHSM.

the active service. Fig. 4.2b shows how the *vPorts* are multiplexed and de-multiplexed on the communication processes of end hosts. In this work, the SDN controller performs this operation with the host multiplexing operation for each connection and synchronized in the mapping table of the end hosts. Algorithm 2 describes the procedure for multiplexing of *vIPs* to *rIP*, and *vPorts* to *rPort* on the SDN-controller, respectively.

4.4.4 Communication Protocols

The communication mechanisms of RHSM are similar to that of our previous work [139] (i.e., presented in Chapter 3) in which the ordinary users (e.g., clients) communicate using domain-name (i.e., host-name) and, the authorized users (e.g., administrators) communicate using the *rIPs* of the server-hosts. The SDN controller installs and modifies the necessary flow-rules of the switches. The client is assumed to be authorized per connection. With

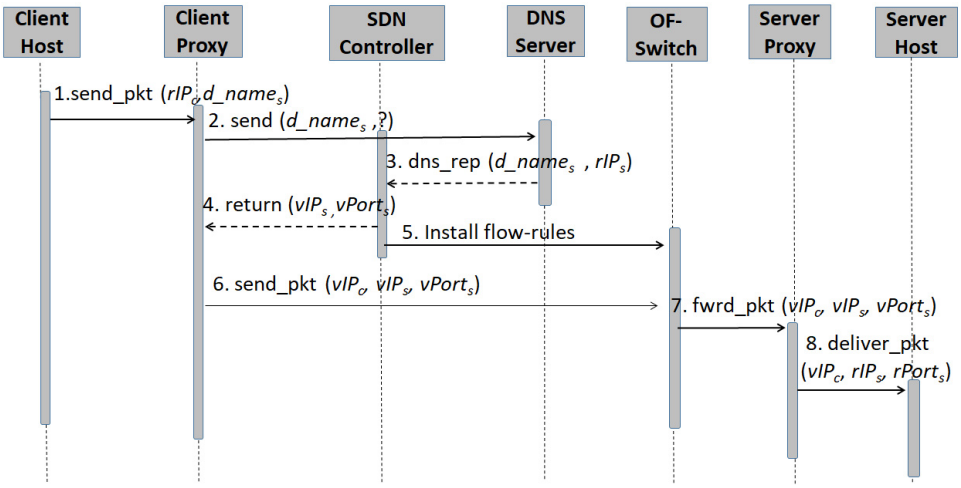


Figure 4.3: Sequence of the communication message flow from a client-host to a server-host using a domain-name.

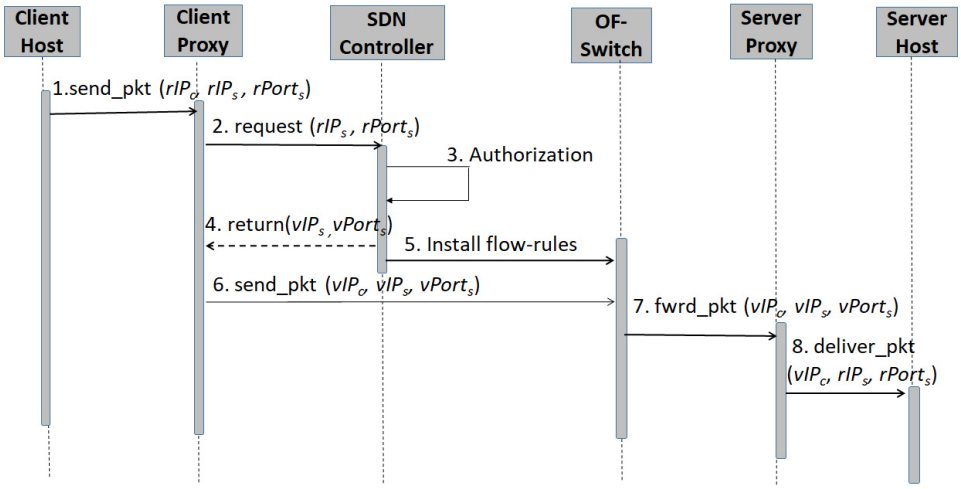


Figure 4.4: Sequence of the communication message flow from a client-host to a server-host using a real IP.

RHSM, *vIPs* are only used for routing while *rIPs and rPorts* are instead hidden within the network. The translation of *virtual-to-real address* (e.g., *vIP-to-rIP* and *vPort-to-rPort*) is performed before delivering the packets at the proxies. Figs. 4.3 and 4.4 show the sequence of the flow of messages from a client-host to a server-host in system with RHSM defense mechanism using a domain-name (or host-name) and *rIP*, respectively.

4.5 Experimental Results and Analysis

In this section, we demonstrate the experimental results that compare the performance of the RHSM and a baseline model with a static network. We investigate the impact of scanning attacks in terms of attacker success probability and the defense cost generated.

4.5.1 Comparing Network Environments

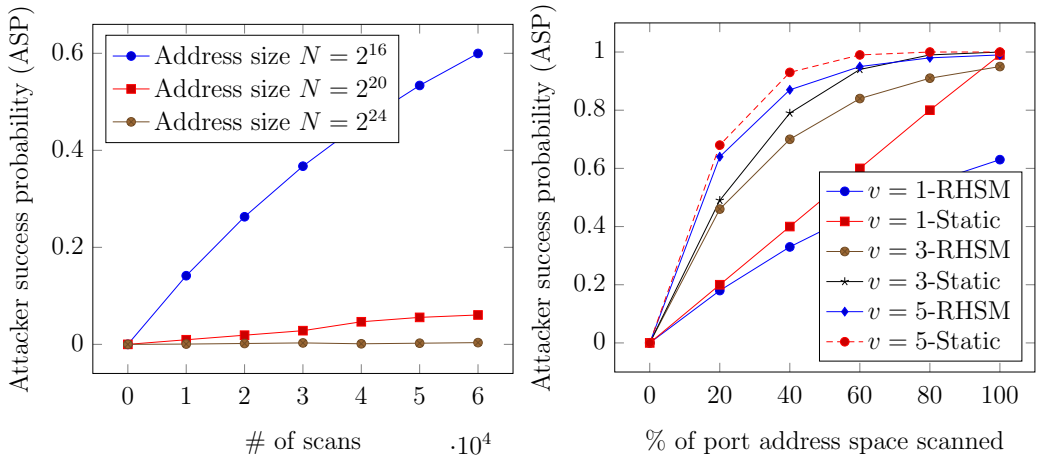
A **static network** is considered as a baseline network to be compared against the proposed RHSM, where the network uses a static network configuration with static IP address and ports assigned to a server-host. The Attacker Success Probability (ASP) to discover at least one server-host or one service running on it in an SDN-based static network with n-hosts, v-vulnerable services, N-virtual IP address size, and m-port address size in k-scans can be computed, respectively, by [23, 139] (i.e., presented in Chapter 3):

$$ASP_h = 1 - \frac{\binom{N-n}{k}}{\binom{N}{k}}, \quad ASP_s = 1 - \frac{\binom{m-v}{k}}{\binom{m}{k}} \tag{4.1}$$

where $k \geq 1$, $N \geq n$ and $m \geq v$.

A dynamic network with RHSM remaps all *vIPs* and *vPorts* of the server-host periodically based on intervals, T_i^{vIP} and T_j^{vPort}, respectively. We assume that it remaps all the *vIPs* addresses and *vPorts* numbers of server-hosts per scanning attempt. An attacker loses any reconnaissance knowledge gained upon every multiplexing event. The ASP to discover at least a server-host or service in an SDN-based network with RHSM scheme is obtained, respectively, by [23, 139]:

$$ASP_h^{rhsm} = 1 - (1 - \frac{n}{N})^k, \quad ASP_s^{rhsm} = 1 - (1 - \frac{v}{m})^k \tag{4.2}$$



(a) ASP to discover a server-host with address size of all private IP addresses of IPv4

(b) ASP under RHSM vs. a static network.

Figure 4.5: Attacker success probability of RHSM versus a static network under a different attack intensity.

4.5.2 Experimental Setup

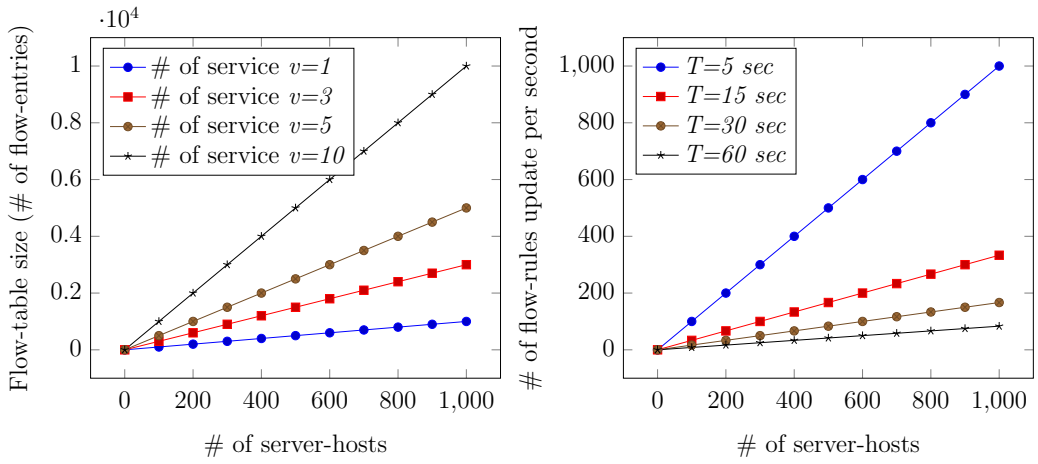
We validate the outperformance of the RHSM, compared to the baseline static network, in terms of ASP and the size of the flow table (i.e., defense cost introduced by the MTD). Recall that the attacker aims to launch scanning attacks while the defense system keeps real IP addresses and port numbers hidden to obfuscate the real IDs of the system components. Hence, the attacker will use a random, virtual IP address or a random, virtual port number in order to identify a target server-host or an active service, respectively. The ASP to discover the target host or the target service is obtained based on Eqs. (4.1) and (4.2) for both the static network and the dynamic network with RHSM, respectively.

4.5.3 Attacker Success Probability Analysis

ASP to discover a server-host: Fig. 4.5a shows the effect of discovering a server-host on ASP under varying the number of scans. In Fig. 4.5a, we investigated how RHSM outperforms the baseline model when the attacker attempts to find a *vIP* ranging for IPv4 Class-A (i.e., 10.0.0.0/8), Class-B (i.e.,

172.16.0.0/12), and Class-C (i.e., 192.168.0.0/16) under varying the number of scans. As shown in Fig. 4.5a, ASP reaches 0.63 when the attackers scan all the private IPs of Class-C (i.e., $2^{16} = 65,536$). This implies that more effort is needed for the attacker to discover the host's IP as the address space increases from IP class C to B. Also, much more effort is needed for the attacker to reach its success from IP class B to class-A. Since the nature of MTD is proactive, we normally do not allow the attacker time to scan the whole network. However, we show this worst-case result for the purpose of demonstrating the outperformance of the RHSM.

ASP to discover services in a server-host: Fig. 4.5b shows the comparative security analysis with respect to ASP to discover vulnerable services in a server-host under a static network vs. a dynamic network with RHSM. We computed the ASP to discover the service in a server-host when the number of vulnerable services varies. In Fig. 4.5b, ASP linearly increments proportional to the number of scans. Under the static network without RHSM, 100% of port scanning attack ($2^{16} = 65,536$) makes 100% of attack success (i.e., $ASP = 1$). On the other hand, under the dynamic network with RHSM, the magnitude of increasing ASP is significantly slowed down even when the attacker scans the whole port address space. Similarly, Fig. 4.5b shows ASP to discover target services in a server-host with a different number of vulnerable services, such as $v = 1, 3$, and 5. As expected, a higher ASP is observed under more vulnerable services. In particular, it is noticeable that the ASP is highly sensitive to the increase of vulnerable services, showing exponential growth with $v = 3, 5$, while showing a linear growth with $v = 1$. In conclusion, the effect of service multiplexing is more pronounced under a smaller number of services vulnerable (i.e., $v \leq 5$). Otherwise, more frequent multiplexing needs to be executed while balancing the overhead incurred by the multiplexing operations.



(a) Flow-table size

(b) Flow-rules update rate

Figure 4.6: Overhead of deploying RHSM on an SDN-based environment.

4.5.4 Overhead Analysis

We use an SDN controller to make a flow entry in the flow table for each host it communicates with, but the end of each session does not change the installed flow rules. The size of a flow table linearly increases in proportion to the number of hosts. Under RHSM, since a host's virtual IP addresses and an active service's port number keep being changed, the SDN controller needs to maintain multiple flow entries and accordingly updates the installed flow rules for each host. Further, since the IP/port multiplexing is performed periodically, a shorter MTD interval leads to more frequent modifications of the flow rules for each host.

The size of a flow table (N_f): For n hosts with each having v-vulnerable services, N_f increases as both hosts and services increase with $O(nv)$. Fig. 4.6a shows that N_f grows as n increases under varying v. This proves that N_f linearly grows as n and v grow.

The frequency of flow rules modifications (N_r): Let $\theta = 1/T$ denote an average multiplexing rate in RHSM. Given $T = T_i^{vIP} = T_j^{vPort}$, RHSM modifies θ times of flow rules per second, leading to $N_r = O(nv\theta)$. In Fig. 4.6b, given a fixed $v = 5$ running on each server-host, we varied the time interval of multiplexing (*e.g.*, $T = 5, 15, 30, 60$ *sec.*) with varying n in x-axis. Shorter

T (e.g., 5 sec.) significantly introduces high overhead showing higher N_r, and vice-versa.

The performance of an SDN controller: Under RHSM, an SDN controller delegates a server program to make the updates on proxies of the end systems via a north-bound interface (i.e., NB-API) in the SDN. The controller makes flow rules updates on the flow table of the SDN switch parallel via *OFPT_FLOW_MOD* operations. Therefore, the overhead of the SDN controller is negligible. Also, there is no cost of address translations to the SDN controller because the end hosts perform the multiplexing/de-multiplexing of *vIPs-rIP* and *vPorts-rPort* on their RHSM proxies.

4.6 Summary

This chapter presented a novel proactive, dynamic SDN-based MTD called RHSM that randomizes hosts (e.g., IP addresses) and services (e.g., port numbers) by employing a multiplexing (or de-multiplexing) technique to confuse attackers and accordingly increases their attack complexity by leveraging the controllability offered by the SDN technology. For the validation of the outperformance of the RHSM, we conducted performance analysis by comparing the RHSM with a baseline model with a static network configuration (i.e., without using RHSM) with respect to the ASP. We analyzed the effect of port address space scanned by attackers under varying a different number of service vulnerabilities. Our results proved that the proposed RHSM could effectively thwart scanning attacks and provide a proactive and resilient security solution by effectively hiding the real identities of both a server-host and its active service in SDN environments. The experimental results proved that RHSM outperformed its static counterpart in terms of the decreased attacker success probability with small additional overhead. Our findings suggest that the proposed RHSM should be deployed more opportunistically to meet the conflicting goals of performance and security.

Chapter 5

Dynamic Security Metrics for MTD

This chapter proposes a new suite of dynamic security metrics to measure the effectiveness of MTD mechanisms in SDNs. The metrics are introduced in Section 5.1. Section 5.2 and Section 5.3 describe the system and attacker models considered in this work respectively. The proposed metrics are presented in Section 5.4. The simulation experiments and results are discussed in Section 5.5. Section 5.6 summarizes the work presented in this chapter.

The main contributions of this chapter are summarized as follows:

- Propose a set of dynamic security metrics that can assess the effectiveness of deployed SDN-based MTD techniques that can timely, dynamically, and adaptively capture a network's security state.
- Develop a comprehensive set of security metrics measuring dynamics in terms of changes made by MTD in network addresses, network topology (or attack paths/routes), and attack success by considering the multi-staged attacks for discovering, exploiting, and compromising a target host.
- Perform extensive simulation experiments to measure the effectiveness of MTD techniques deployed in SDNs, identifying key factors that

significantly influence the performance of MTD based on the proposed security metrics.

5.1 Introduction

MTD is a game-changing cybersecurity strategy that creates uncertainty of network configurations introducing higher hurdles for attackers [77]. The key purpose of MTD is to change attack surface [106] to increase unpredictability for attackers. The technology of software-defined networks (SDNs) has emerged as a platform for MTD techniques to be efficiently and effectively deployed. A variety of SDN-based MTD techniques have been developed [30, 81, 103] as proactive and dynamic security solutions. Several assessment methods for MTD techniques [64, 166, 176] have been proposed. However, their application is limited to particular attack scenarios or models. For instance, a system risk metric can be best suited to assess the effectiveness of virtual machine shuffling MTD, but not for assessing the effectiveness of network and address shuffling based MTD. Also, timeliness-based MTD frequently and periodically changes a network's configurations per time interval of T. This may miss capturing a security state at $T + 1$ because the status check is performed every T time units. In this regard, the existing security metrics do not fully capture the dynamics of a network in terms of the changes made in the attack surface after MTD being executed. In this chapter, we propose a new suite of dynamic security metrics to measure the effectiveness of SDN-based MTD techniques in which they can timely and adaptively capture dynamic security states of the network over time. These metrics estimate *variability* to keep track of the changing patterns of network configurations or status (e.g., IP addresses, services, paths, connectivity, vulnerabilities) per MTD interval.

5.2 System Model

In this work, we consider the SDN-based system model. The SDN-based networked system consists of end hosts, SDN switches (i.e., OF-switches), and SDN-controller (e.g., Ryu [133]) which are communicating over a secure channel via an open-flow (OF) protocol [107]. The SDN technology decouples a network control plane with a data-forwarding plane in order to provide high flexibility and programmability [136], which can effectively abstract the underlying infrastructure for high applicability and service provision [88]. The programmable interfaces afforded by the SDN can be comforted to deploy and analyze MTD techniques. The SDN controller implements the MTD technique extending its functionalities and controls in a centralized manner [81, 139]. An example network based on this system model is presented in Fig 5.1.

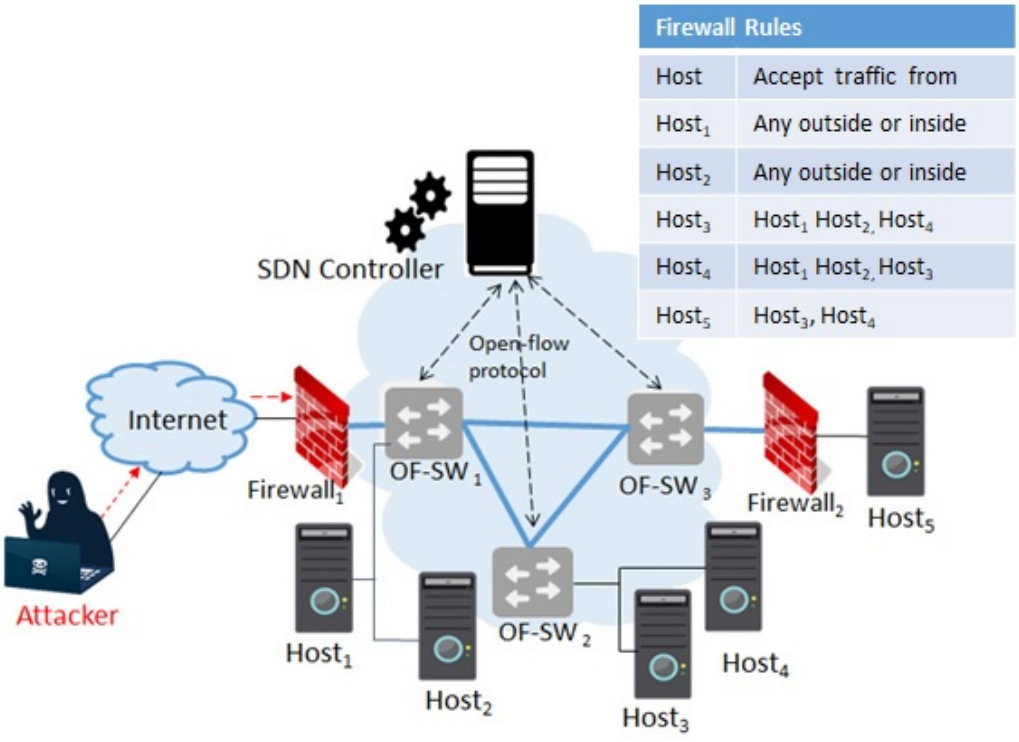


Figure 5.1: An example of the SDN-based network environment for deploying the MTD mechanism.

5.3 Attacker Model

In this work, we devise a three-stage attack model based on Cyber Kill Chain (CKC) model [70], consisting of the following three phases:

1. *Discovery* is a reconnaissance phase involving the identification or selection of potential victim targets (e.g., hosts or services) and the collection of the security-related information (e.g., vulnerabilities or reachability) via scanning on the target system.

2. *Exploitation* is a phase of the attacker attempting to exploit applications or operating system vulnerabilities (e.g., cross-site scripting or remote code execution) of the target system.

3. *Attack* is a phase after the previous two phases are successfully performed. Hence, in this phase, the attacker launches an attack to compromise the target system.

The attacker is assumed to have a certain level of knowledge towards a target system and its defense capability as:

- The attacker is aware of the network address space and performs reconnaissance on targeted network adapting random scanning strategy.
- The attacker is located outside the network (i.e., outside attackers).
- The attacker assumes each host has at least one vulnerability and can exploit it.
- Exploiting a host's vulnerability means granting the root privilege of the host.
- The attacker uses scanning tools (e.g., *Nmap* [101] or *Nessus* [111]) to discover all hosts and vulnerabilities on the targeted network.
- Each attacker is aware of an MTD technique running on the target network by shuffling IP and network addresses.

5.4 Proposed Metrics

In this section, we describe our proposed security metrics aiming to effectively capture the dynamics introduced by an SDN-based MTD that can enhance system security. These metrics are capable of capturing the dynamics of the network state information over time. The network state information comprises host or network configuration information (e.g., IP addresses, TCP/UDP ports, OS, vulnerabilities, network reachability-paths, or routes). These metrics compute *variability* that keeps track of changing patterns (i.e., variations) on the configuration information based on the observations of the network's states over time. We categorize our metrics into three classes (i.e., address-based, path-based, and attack stage-based) based on the information used or attack model consideration.

5.4.1 Network and Host Address-based Metrics

In this section, we propose two networks address information-based security metrics: network address variability (NAV) and host address variability (HAV) to assess the effectiveness of MTD. These metrics compute the changing pattern of the IP addresses over time.

5.4.1.1 Network address variability (NAV)

This measures the change in the network (IP) addresses over time. A set of currently used IP addresses of the hosts in the network at time t_k is defined by $\mathbb{NS}_{t_k} = \{h_i.IP_{t_k} \mid h_i.IP_{t_k}$ is a currently used IP of $h_i \in \mathbb{H}\}$. An IP-shuffling MTD technique constantly changes the IP addresses of the hosts in every T time interval (i.e., sec.). This dynamic changes of the IP addresses can be captured by the network states (i.e., $\mathbb{NS}_{t_k}$) at a different time point (i.e., t_k). The variability of the two consecutive network states is computed by:

$$NAV_{t_k} = \frac{|\mathbb{NS}_{t_k} - \mathbb{NS}_{t_{k-1}}|}{|\mathbb{NS}_{t_k}|} \ , \tag{5.1}$$

where "−" is the set difference operations and "$|\cdot|$" is the cardinality of the set. Eq. (5.1) is the number of address changes on $\mathbb{NS}_{t_k}$ normalized by its size. NAV_{t_k} is estimated based on how the network state changing over time and rangeed in $[0, 1]$. For instance, if the network state remains unchanged, $NS_{t_k} - NS_{t_{k-1}} = \phi$ holds then $NAV_{t_k} = 0$; if it has partially changed, $NS_{t_k} - NS_{t_{k-1}} \subset NS_{t_k}$ holds then $0 < NAV_{t_k} < 1$; and if it has completely changed, $NS_{t_k} - NS_{t_{k-1}} = NS_{t_k}$ holds then $NAV_{t_k} = 1$. The average network address variability, or NAV, for a given scanning time window $[t_1, t_m]$ is:

$$NAV = \frac{\sum_{k=2}^{m} NAV_{t_k}}{m-1} \ . \tag{5.2}$$

Based on Eq. (5.1), we can easily infer the NAV based on two consecutive observable network states for the given duration $[t_1, t_m]$. We can compute the NAV for multiple observable network states by finding the union of a current set of network states ($\mathbb{NS}_{t_k}$) and a set of all previous network states (i.e., $\mathbb{NS}_{t_{k-1}}, \mathbb{NS}_{t_{k-2}}, \cdots, \mathbb{NS}_{t_1}$). The overall NAV at t_k is:

$$NAV^{U}_{t_k} = \frac{|\mathbb{NS}_{t_k} - (\cup_{i=1}^{k-1} \mathbb{NS}_{t_i})|}{|\mathbb{NS}_{t_k}|} \ . \tag{5.3}$$

Like NAV_{t_k}, $NAV^{U}_{t_k} \in [0, 1]$, indicating 0 for low variability and 1 for high variability.

5.4.1.2 Host address variability (HAV)

NAV measures how the deployed MTD creates the address diversity in a network, representing address variability at a network-level rather than at a host-level. Therefore, to measure the host-level variability, we devise a metric measuring the changing pattern of *IP* addresses assigned to each host individually over time. Given $\mathbb{HS}_{t_k}$, a set of currently used IP address(es) of the host in the network at time t_k, the variability of each host's address at t_k

is:

$$HAV_{t_k} = \frac{|\mathbb{HS}_{t_k} - \mathbb{HS}_{t_{k-1}}|}{|\mathbb{HS}_{t_k}|} \ . \tag{5.4}$$

The HAV for a given scanning time window $[t_1, t_m]$ is:

$$HAV = \frac{\sum_{k=2}^{m} HAV_{t_k}}{m-1} \ . \tag{5.5}$$

Similar to $NAV^U_{t_k}$, we can compute $HAV^U_{t_k}$ representing the variability of currently assigned IP address with union of all the previous states for multiple observable states as:

$$HAV^U_{t_k} = \frac{|\mathbb{HS}_{t_k} - (\cup_{i=1}^{k-1} \mathbb{HS}_{t_i})|}{|\mathbb{HS}_{t_k}|} \ . \tag{5.6}$$

Both NAV and HAV are in the range of $[0, 1]$, where higher values indicate higher variabilities introduced by MTD.

5.4.2 Path-based Metrics

In this section, we propose path-based security metrics that measure MTD effectiveness in creating uncertainty on attack paths when shuffling IP addresses or network topology (e.g., connecting/disconnecting links between hosts). These dynamic network changes affect hosts' reachability and accordingly change attack paths. We use the Temporal Hierarchical Attack Representation Model (T-HARM) [47] to capture the changes of attack paths. T-HARM is a two-layer security model that captures a host's temporal reachability on the upper layer and vulnerability at the lower layer using the attack graph (AG) and the set of attack trees (ATs), respectively.

5.4.2.1 Attack path variability (APV)

This metric measures the changes in attack paths to capture the effect of deploying MTD. The MTD shuffling makes the network topology frequently changed. We represent this dynamics of the network over time (i.e., at a

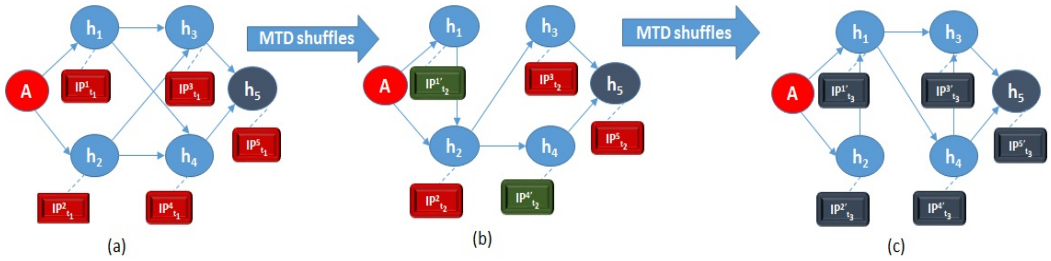


Figure 5.2: Network graph with reachability and address information of the example network: (a) at time t_1, (b) at time t_2, and (c) at time t_3.

different time point t_k) with AG_{t_k} and compute all possible attack paths in every t_k. For instance, Fig. 5.2 shows a network reachability graph, that can derive attack paths, at three different time points (i.e., t_1, t_2, and t_3). The set of all possible attack paths at t_k is described by:

$$\mathbb{AP}_{t_k} = \{ap_{t_k} \mid ap_{t_k} \in AG_{t_k}\} \tag{5.7}$$

where ap_{t_k} is an attack path from an attacker to a target at t_k. Similar to the computation of NAV in Eq. (5.1), we estimate the instantaneous APV at t_k, denoted by APV_{t_k}, based on $\mathbb{AP}_{t_k}$ and APV_{t_k} is given by:

$$APV_{t_k} = \frac{|\mathbb{AP}_{t_k} - \mathbb{AP}_{t_{k-1}}|}{|\mathbb{AP}_{t_k}|} \ . \tag{5.8}$$

The overall APV for the duration of the given time window $[t_1, t_m]$ is:

$$APV = \frac{\sum_{k=2}^{m} APV_{t_k}}{m-1} \tag{5.9}$$

where APV is ranged as a real number in $[0, 1]$.

We discuss how APV can be used in the example network (i.e.,Fig. 5.1) based on Example 1 below. The network consists of 5-hosts (i.e., h_1, h_2, h_3, h_4, and h_5) where h_5 is the target host. An attacker A is located outside the network and attempts to attack the target host (i.e., h_5) by discovering an attack path and exploiting the vulnerabilities of those hosts on the attack path. The deployed MTD mechanism shuffles the network topology to disrupt

the attacker's activities. This metric captures such dynamics introduced to the network due to the MTD operations and computes variations in terms of attack paths at different time points. Figs. 5.2 (a), (b), (c) show the representations of the example network at t_1, t_2, t_3, respectively. We compute all possible attack paths and their variations over time.

Example 1. A set of all possible attack paths at time t_1, $\mathbb{AP}_{t_1}$= $\{(A, h_1, h_3, h_5), (A, h_1, h_4, h_5), (A, h_2, h_4, h_5), (A, h_2, h_3, h_5)\}$, at time t_2, $\mathbb{AP}_{t_2}$= $\{ (A, h_1, h_2, h_4, h_5), (A, h_1, h_2, h_3, h_5), (A, h_2, h_4, h_5), (A, h_2, h_3, h_5) \}$. APV at time t_2, $APV_{t_2} = \frac{|\mathbb{AP}_{t_2} - \mathbb{AP}_{t_1}|}{|\mathbb{AP}_{t_2}|} = \frac{2}{4} = 0.5$. Similarly,$APV$ at time t_3, $APV_{t_3} = \frac{6}{6} = 1.0$. Now, we obtain APV on time window $[t_1, t_3]$ (i.e., $m = 3$), $APV = \frac{0.5+1.0}{3-1} = 0.75$. This shows that deploying MTD technique invalidates 75% of the attack paths on scanning time window $[t_1, t_3]$.

5.4.2.2 Attack path variability with IP-shuffling (APVIS)

APV metric is designed to capture the impact of the dynamics introduced to the network topology on attack paths due to network shuffling. APV does not reflect the effect of IP-shuffling. To estimate the dynamics of the network introduced by network address shuffling MTDs, we propose a metric called APV with IP-shuffling, namely APVIS. This metric measures the impact of IP shuffling MTD on attack paths based on the rationale that more IP changes disrupt attackers more with high attack complexity. We compute the attack path variability based on the changes made in an IP address of each host on the path. We obtain a set of attack paths with IP-shuffling at t_k by:

$$\mathbb{APIS}_{t_k} = \{ap_{t_k} \mid ap_{t_k} \in AG_{t_k}\} \tag{5.10}$$

We define a binary variable $b^i_{t_k}$ to keep track of the IP changes of each host, $h_i \in \mathbb{H}$, at two different time points t_k and t_{k-1} by:

$$b^i_{t_k} = \begin{cases} 1 & \text{if } h_i.IP_{t_k} \neq h_i.IP_{t_{k-1}} \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

The attack paths can be computed using $b^i_{t_k}$ for each h_i on $ap^j_{t_k} \in \mathbb{APIS}_{t_k}$. h_i is denoted by h'_i if its IP address has been changed (i.e., $b^i_{tk} = 1$). The *APVIS* at t_k can be computed by:

$$APVIS_{t_k} = \frac{|\mathbb{APIS}_{t_k} - \mathbb{APIS}_{t_{k-1}}|}{|\mathbb{APIS}_{t_k}|} \ . \tag{5.12}$$

Now we compute the overall *APVIS* for $[t_1, t_m]$ by:

$$APVIS = \frac{\sum_{k=2}^{m} APVIS_{t_k}}{m-1} \ . \tag{5.13}$$

We show an example of using the metrics as below based on the network in Fig. 5.1. In this example, both a network topology and host IP addresses are changed by MTD.

Example 2. The computation of the set of all possible attack paths at different time points t_1, t_2, and t_3 with network and/or IP shuffling for the example network represented in Fig 5.2 is based on Eq. (5.10) and Eq. (5.11). $\mathbb{APIS}_{t_1}$= {(A, h_1, h_3, h_5), (A, h_1, h_4, h_5), (A, h_2, h_4, h_5), (A, h_2, h_3, h_5) }; $\mathbb{APIS}_{t_2}$= { (A, h'_1, h_2, h'_4, h_5), (A, h'_1, h_2, h_3, h_5), (A, h_2, h'_4, h_5), (A, h_2, h_3, h_5) }; and $\mathbb{APIS}_{t_3}$ ={ (A, h'_1, h'_3, h'_5), (A, h'_1, h'_4, h'_5), (A, h'_1, h'_4, h'_3, h'_5), (A, h'_2, h'_1, h'_3, h'_5), (A, h'_2, h'_1, h'_4, h'_5), (A, h'_2, h'_1, h'_4, h'_3, h'_5) }. Based on Eq. (5.12), we obtain $APVIS_{t_2}$ =$\frac{3}{4}$= 0.75, and $APVIS_{t_3}$= $\frac{6}{6}$= 1.0. Therefore, APVIS on time window $[t_1, t_2] = \frac{0.75+1.0}{3-1} = 0.875$. This indicates that the deployed MTD technique disrupts the attacker by invalidating 87.5% of addresses and/or paths information collected by the attacker during the time window $[t_1, t_3]$.

5.4.2.3 Shortest attack path variability (SAPV)

The shortest path is a common graphical security metric that represents the length of the smallest attack paths from an attacker's initial state to the goal state. This metric estimates the changes on shortest attack paths over time. A set of the shortest attack path at t_k is estimated by:

$$\mathbb{SAP}_{t_k} = \{ap_{t_k} \mid \min_{ap_{t_k} \in \mathbb{AP}_{t_k}} |ap_{t_k}|\} \tag{5.14}$$

The $SAPV$ at t_k is estimated by:

$$SAPV_{t_k} = \frac{|\mathbb{SAP}_{t_k} - \mathbb{SAP}_{t_{k-1}}|}{|\mathbb{SAP}_{t_k}|} \ . \tag{5.15}$$

The overall $SAPV$ for $[t_1, t_m]$ is calculated as:

$$SAPV = \frac{\sum_{k=2}^{m} SAPV_{t_k}}{m-1} \ . \tag{5.16}$$

Now we discuss how the SAPV metric is used in Example 3 for the network graphs in Fig. 5.2. The computation of the SAPV is similar to that of the APV metric. However, its calculation is based on the shortest attack paths which are more potential attack paths chosen by the attacker.

Example 3. We have the shortest attack paths at t_1, $\mathbb{SAP}_{t_1}$ = {(A, h_1, h_3, h_5), (A, h_1, h_4, h_5), (A, h_2, h_4, h_5), (A, h_2, h_3, h_5)}; at t_2, $\mathbb{SAP}_{t_2}$ = {(A, h_2, h_4, h_5), (A, h_2, h_3, h_5)}; and at t_3, $\mathbb{SAP}_{t_3}$= {(A, h_1, h_3, h_5), (A, h_1, h_4, h_5)}. Now, $SAPV_{t_2} = \frac{0}{2} = 0.0$; $SAPV_{t_3} = \frac{2}{2} = 1.0$, The overall SAPV= $\frac{0+1}{3-1} = 0.5$. $SAPV_{t_2} = 0.0$ means there is no variation on SAP at time t_2, implying that shortest attack paths remain unchanged (i.e., all of the SAP at time t_2 also exist at time t_1), so the attacker can reach the target host using the shortest paths and perform attack on it. This shows that there is no effect on the SAP of the deployed MTD at time t_2. Similarly, $SAPV_{t_3} = 1.0$ means a perfect variation on the SAP at time t_3 (i.e., none of the shortest attack paths at time

t_2 exist at time t_3). $SAVP = 0.5$ indicates that the deployed MTD invalidated 50% of the shortest attack paths obtained during scanning time window $[t_1, t_3]$.

5.4.2.4 Shortest attack path variability with IP-shuffling (SAPVIS)

The set of the shortest attack paths with IP-shuffling at t_k is given by $\mathbb{SAPIS}_{t_k} = \{ap_{t_k} \mid \min_{ap_{t_k} \in \mathbb{APIS}_{t_k}} |ap_{t_k}|\}$. The $SAPVIS$ at t_k is computed by:

$$SAPVIS_{t_k} = \frac{|\mathbb{SAPIS}_{t_k} - \mathbb{SAPIS}_{t_{k-1}}|}{|\mathbb{SAPIS}_{t_k}|} \ . \tag{5.17}$$

The overall $SAPVIS$ for $[t_1, t_m]$ is:

$$SAPVIS = \frac{\sum_{k=2}^{m} SAPVIS_{t_k}}{m-1} \ . \tag{5.18}$$

Note that unlike APVIS, SAPVIS is computed based on the shortest attack paths.

5.4.2.5 Variability of the number of attack paths (VNAP)

This metric refers to the number of attack paths for an attacker to reach a target host. If the network topology is randomly shuffled per interval, it changes the number of attack paths available. When more paths are available, the attacker can have various alternatives to reach the target host. Therefore, computing the variability of NAP per T time interval can be used to assess the effectiveness of the MTD techniques over time. The NAP at t_k can be computed by:

$$NAP_{t_k} = |\mathbb{AP}_{t_k}| \tag{5.19}$$

The $VNAP$ at t_k can be obtained as:

$$VNAP_{t_k} = \min\left[\frac{NAP_{t_{k-1}}}{NAP_{t_k}}, 1\right] \ . \tag{5.20}$$

The overall $VNAP$ for $[t_1, t_m]$ can be represented by:

$$VNAP = \frac{\sum_{k=2}^{m} VNAP_{t_k}}{m-1}, \tag{5.21}$$

where VNAP is ranged in $[0, 1]$ as a real number where a smaller VNAP (i.e., < 1) increases the number of attack paths available. How the VNAP is computed in the example network is discussed in Example 4.

Example 4. Based on Eq. 5.19, we compute NAP at different time points, t_1, t_2, and t_3 using the network graph of the example network. We have, $NAP_{t_1} = |\mathbb{AP}_{t_1}| = 4$, $NAP_{t_2} = |\mathbb{AP}_{t_2}| = 4$, $NAP_{t_3} = |\mathbb{AP}_{t_3}| = 6$. Now, VNAP at t_2, $VNAP_{t_2} = \min(\frac{4}{4}, 1) = 1.0$; similarly, VNAP at t_3, $VNAP_{t_3} = \min(\frac{4}{6}, 1) = 0.66$. We obtain overall VNAP $= \frac{1.0+0.66}{3-1} = 0.83$. It means that there is a negative impact of the MTD shuffling (i.e., the attacker can find more attack paths available) since NAP remains unchanged at t_2 (i.e., $NAP_{t_2} = 4, VNAP_{t_2} = 1.0$) and it increases to 6 at time t_3 (i.e., two more paths, $NAP_{t_3} = 6$).

5.4.3 Attack Stage-based Success Metrics

In this section, we propose a set of security metrics that measure an attacker's success probability where the attacker performs a multi-staged attack by discovering, exploiting, and finally compromising a target system.

5.4.3.1 Scan success probability (SSP)

An attacker's SSP to discover h_i's information (i.e., $h_i \in \mathbb{H}$), such as IP, services, and/or vulnerabilities in r scans at t_k can be defined by:

$$SSP_{h_i} = 1 - \prod_{j=1}^{r}(1 - p_j) \ , \tag{5.22}$$

where p_j is a success probability in each scan and $r \leq N$. For a sufficiently large N (i.e., $N \to \infty$) with equal probability in each scan (i.e., $p_j = p, \forall j$), the SSP

in scanning all addresses (i.e., $r = N$) is $SSP_{h_i} = 1-(1-\frac{1}{N})^N = 1-e^{-1} \approx 0.63$.

5.4.3.2 Exploit success probability (ESP)

The ESP for the attacker to exploit $h_i \in \mathbb{H}$ depends on whether h_i's vulnerability is exploitable or not. Here we use the lower layer of the T-HARM [47] (i.e., the dynamic attack trees (ATs) which have both the AND and OR gates) to calculate the ESP. The detailed mapping of the gates and constructions of the ATs can be found in [11]. The formula for the ESP, where v_j^i is the vulnerability in h_i, is given by:

$$ESP_{h_i} = \begin{cases} \prod\limits_{v_j^i} p(v_j^i), & \forall v_j^i \in AND, \quad v_j^i \in h_i \\ 1 - \prod\limits_{v_j^i} (1 - p(v_j^i)), & \forall v_j^i \in OR, \quad v_j^i \in h_i \ . \end{cases} \tag{5.23}$$

where $p(v_j^i)$ is the probability to exploit h_i's vulnerability j, v_j^i in h_i. The $p(v_j^i)$ is obtainable from the Common Vulnerability Scoring System (CVSS) [39]'s base score (bs_j^i), and can be calculated as $p(v_j^i) = bs_j^i / VS_{max}$, where $bs_j^i = (0.6 \times \text{Impact}_j^i + 0.4 \times \text{Exploitability}_j^i - 1.5) \times f(\text{Impact}_j^i)$ and VS_{max} is the maximum vulnerability score set to 10.

5.4.3.3 Compromise success probability (CSP)

The CSP is a likelihood of an attacker's success in compromising a target $h_i \in \mathbb{H}$ of the targeted system. As discussed in our threat model in Section 5.3, $h_i \in \mathbb{H}$ is considered being attacked if the attacker successfully gathered all the required information of h_i and exploited v_j^i (i.e., v_j^i in h_i). Therefore, CSP towards h_i is based on the product of SSP_{h_i} and ESP_{h_i} and is computed by $CSP_{h_i} = SSP_{h_i} \times ESP_{h_i}$. The overall CSP on an attack path $ap_{t_k}^j \in \mathbb{AP}_{t_k}$ is a product of all CSP_{h_i} on $ap_{t_k}^j$ is given by:

$$CSP_{ap_{t_k}^j} = \prod_{h_i \in ap_{t_k}^j} CSP_{h_i}, \quad ap_{t_k}^j \in \mathbb{AP}_{t_k} \ . \tag{5.24}$$

Table 5.1: Security and vulnerability information of the hosts in the example network.

Host (h_i)	**Vul.** (v_j^i)	**CVE-ID**	**CVSS** bs_j^i	ESP_{h_i}	SSP_{h_i}
h_1	v_1^1	CVE-2018-11349	6.8	0.68	0.25
h_2	v_2^2	CVE-2018-1111	7.9	0.79	0.37
h_3	v_3^3	CVE-2018-8304	7.1	0.71	0.50
h_4	v_4^4	CVE-2018-2680	5.1	0.51	0.63
h_5	v_5^5	CVE-2018-2579	4.3	0.43	0.67

The CSP on $\mathbb{AP}_{t_k}$ can be maximum $CSP_{ap^j_{t_k}}$ of all attack paths on $\mathbb{AP}_{t_k}$, which is obtained by:

$$CSP_{t_k} = \max_{ap^j_{t_k} \in \mathbb{AP}_{t_k}} CSP_{ap^j_{t_k}} \ . \tag{5.25}$$

The above CSP estimation is based on scanning success (i.e., an attacker's effort), exploit success (i.e., vulnerabilities) and attack paths (i.e., network topology shuffled by MTD) information at t_k. We can compute the CSP for $[t_1, t_m]$ as:

$$CSP = \frac{\sum_{k=1}^{m} CSP_{t_k}}{m} \ . \tag{5.26}$$

Example 5 below shows how the proposed CSP is used in Fig. 5.1, given a network's host and reachability information in Table 5.1 and Fig. 5.2, respectively.

Example 5. $\mathbb{AP}_{t_1} = \{ap^1_{t_1} = (A, h_1, h_3, h_5),\ ap^2_{t_1} = A,\ h_1, h_4, h_5),\ ap^3_{t_1} = (A, h_2, h_4, h_5),\ ap^4_{t_1} = (A, h_2, h_3, h_5)\}$. $CSP_{ap^1_{t_1}} = (SSP_{h_1} \times ESP_{h_1}) \times (SSP_{h_3} \times ESP_{h_3}) \times (SSP_{h_5} \times ESP_{h_5}) = (0.68 \times 0.25) \times (0.71 \times 0.5) \times (0.43 \times 0.67) = 0.017$. Similarly, $CSP_{ap^2_{t_1}} = 0.015$, $CSP_{ap^3_{t_1}} = 0.026$, and $CSP_{ap^4_{t_1}} = 0.029$. CSP to attack host h_5 at time t_1 on attack paths $\mathbb{AP}_{t_1}$, $CSP_{t_1} = 0.029$. Similarly, the CSP to attack h_5 on $\mathbb{AP}_{t_2}$ and $\mathbb{AP}_{t_3}$ is 0.029, and 0.017, respectively. CSP with scanning time window $[t_1, t_3]$ is 0.075.

5.5 Experimental Results & Analysis

In this section, we conducted simulation experiments for evaluating the impact of key parameters on the proposed dynamic security metrics. The obtained results are presented.

5.5.1 Experimental Setup

We created an initial network topology based on a random graph consisting of a given number of hosts and randomly assigned an IP address. Each host's IP is dynamically and randomly changed per MTD time interval T $(= t_k - t_{k-1})$. Similarly, we applied a network topology shuffling MTD by randomly shuffling links (i.e., edges) in the network. The host's information (e.g., IP, vulnerabilities) or reachability information (e.g., paths, links) are captured per T interval and their changes are measured based on a suite of the proposed metrics. For sensitivity analysis, we varied the number of hosts, the number of shuffling links, and the scan success rate with known vulnerabilities on the hosts.

5.5.2 Results & Analysis

Evaluation based on NAV: Fig. 5.3 (a) presents the simulation results using the NAV metric under varying the number of hosts over time. NAV shows little fluctuation over T. Also, NAV is higher under a small-sized network than that under a large-sized network due to a slim chance of overlapping two network states under a small-sized network. NAV in a scanning time window $[0, 150]$ (in sec.) is 0.8 for $n = 50$, indicating that the IP-shuffling mechanism invalidates 80% reconnaissance on scanning the network in the time window. Fig. 5.3 (b) shows the effect of varying the number of hosts on NAV with respect to different address sizes. Notice that NAV gradually decreases as the number of hosts increases because the same number of addresses available needs to be mapped to a larger number of hosts. However, NAV increases considerably as

the size of the available address space (N) increases because higher N generates more diverse addresses with a larger pool of the addresses available.

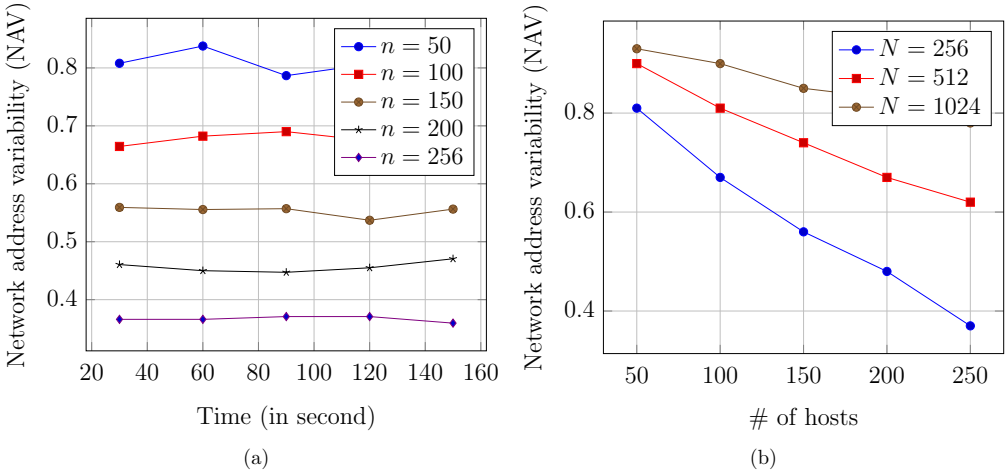


Figure 5.3: NAV measured with respect to (a) varying time for the different the number of hosts (n) with $T = 30s$ and $N = 2^8$; and (b) varying n with different N.

Evaluation based on APV: Fig. 5.4 (a) shows the effect on the variability of attack paths when varying the number of links swapping under a different network size (n). APV jumps from 0 to 0.55 with $n = 10$ when swapping two links (i.e., one swap); and APV increases sharply as the number of swaps increases and reaches to 1 (i.e., 100% variability) in 10 swaps. Similarly, Fig. 5.4 (b) depicts the effect of % of links shuffled under a different network size, n, on APV. When only a small % of links are shuffled, the number of attack paths can change significantly, leading to a greater APV.

Evaluation based on CSP: Fig. 5.5 (a) investigates how the network size, n, affects CSP under different SSP. CSP decreases gradually as n grows due to decreasing ESP. Fig. 5.5 (b) examines the effect of varying n across different % of successful scans. 0% successful scans indicate that an attacker is completely unaware of any configurations towards a target system while 100% successful scans imply an attacker being fully aware of the target system configurations, respectively. CSP increases as more successful scans are performed but slightly decreases as the network size increases because a larger network contributes

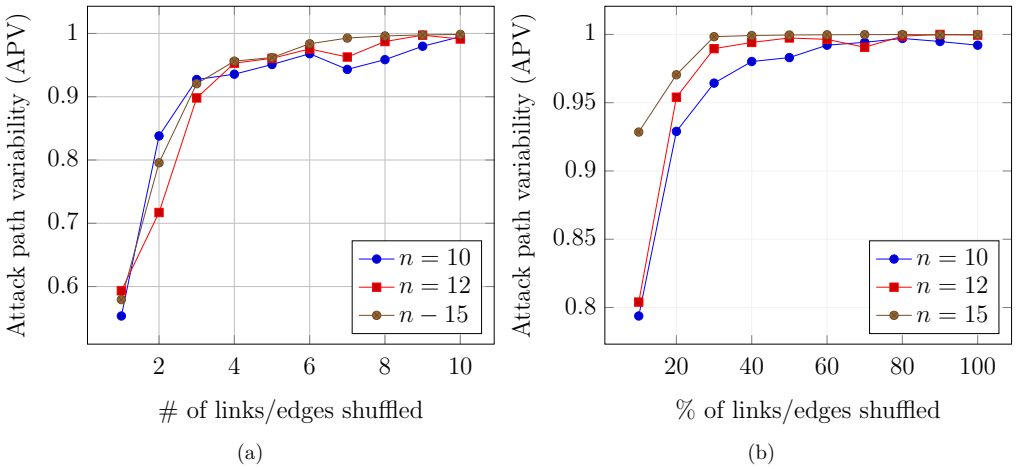


Figure 5.4: APV measured under varying the number of hosts (n) in a network with respect to (a) # of links shuffled; and (b) % of links shuffled.

to generating more attack paths requiring more efforts to identify/exploit vulnerabilities, resulting in greater attack cost.

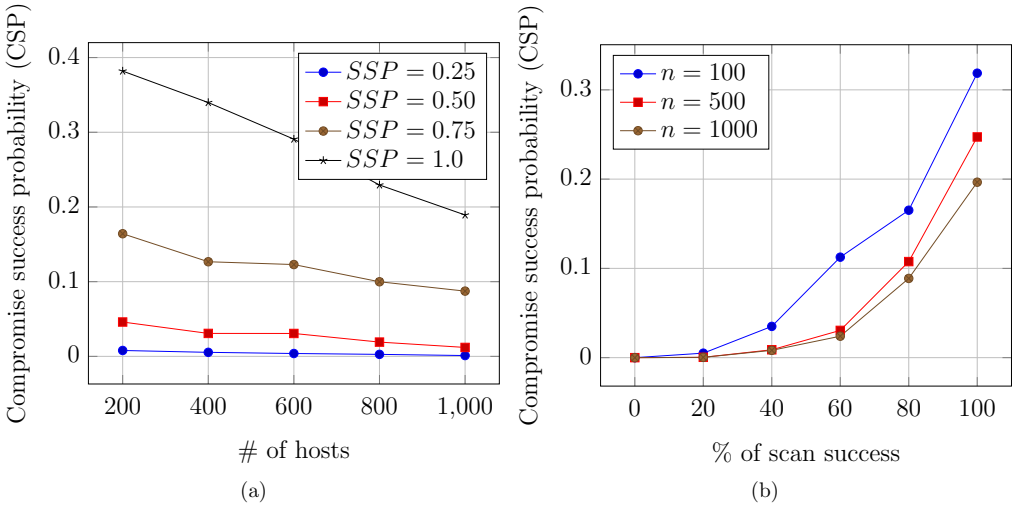
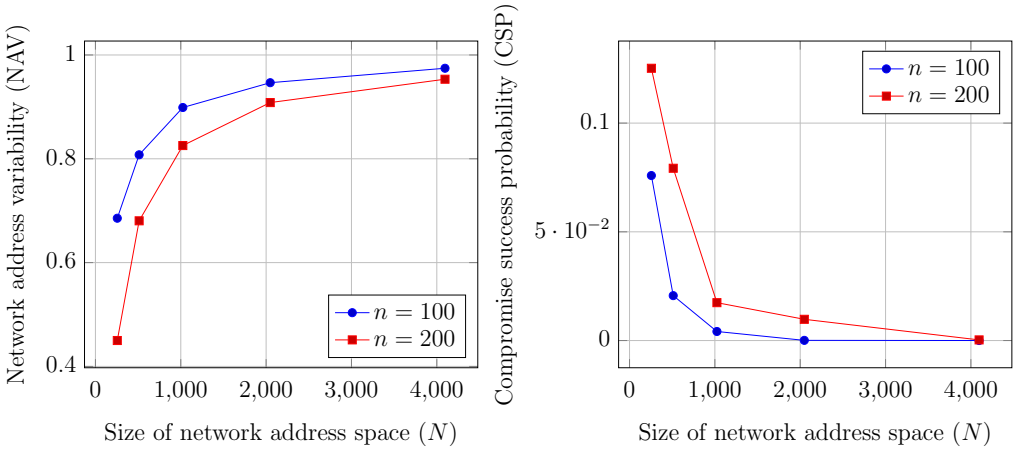


Figure 5.5: CSP measured with respect to (a) varying the number of hosts (n) with different SSP; and (b) varying % of successful scans with different network size (n).

Relationship between NAV and CSP: Figs. 5.6 (a) and (b) show NAV and CSP under two network sizes (i.e., # of hosts, n, is set to 100 or 200), respectively. From this result, we can clearly see the relationship between NAV and CSP: Higher NAV is well aligned with lower CSP, strongly suggesting

that higher network address variability introduced by shuffling-based MTD can reduce the attack success probability more and vice-versa.



(a) NAV (b) CSP

Figure 5.6: Relationship between NAV and CSP metrics.

5.6 Summary

This chapter presented a suite of dynamic security metrics for measuring the effectiveness of MTD techniques that shuffle a network topology and IP addresses. The metrics measuring the variability of network and host addresses properly capture the degree of an attacker being disrupted by a deployed MTD in identifying a targeted network and host's information. The results showed that higher variability can be achieved under networks with larger address space size and smaller network size. Similarly, the attack path-based metrics captured the dynamic change of a network due to MTD link shuffling and measured how the MTD creates uncertainty on attack paths. Based on our experimental results, we showed high path variability significantly decreases attack success probability. The proposed attack stage-based success metrics can be used to assess the MTD techniques when an attacker is in a various stage of its attack. We devised CSP to capture an attacker compromising a target host by considering both *SSP* and *ESP* along with attack path's information. We proved high CSP occurs with high SSP and ESP.

Chapter 6

Discussions and Future Work

The research presented in this thesis addresses all the research questions outlined in Chapter 1 by developing flexible random virtual IP multiplexing (e.g., FRVM), random host and service multiplexing (e.g., RHSM) as novel proactive MTD approaches, and dynamic security metrics for measuring the effectiveness of the deploying MTD mechanisms (e.g., MTD metrics) in SDNs. However, several assumptions are considered, which limit the scope of the work and require future research to increase the scope and impact of the proposed approaches. In this chapter, we discuss the limitations of the proposed approaches and point out possible future work.

6.1 Addressing the Research Questions

In this thesis, we use development of MTD mechanisms and security metrics approach to answer the following research questions outlined in Chapter 1:

- Q1: How can we shuffle virtual IP addresses of server hosts in the SDN?
- Q2: How can we protect the SDN with the existence of the vulnerable services on the server systems in the SDN?
- Q3: How can we measure the effectiveness of moving target defenses in the SDN?

Addressing the research question Q1: To address the research question Q1, a flexible random virtual IP multiplexing (named FRVM) is developed and presented in Chapter 3. FRVM is a novel SDN-based MTD mechanism that is developed to thwart network reconnaissance and scanning attacks by effectively shuffling of the virtual IP addresses. It enables a host to have multiple virtual IP addresses that are multiplexed to a real IP address of the host and frequently changes the virtual IP addresses. SDN-based architecture, mapping algorithms, and communication protocols are developed and implemented in a virtualized network created using *Mininet* [91] emulator. The comparative analysis of the performance is evaluated via simulation experiments using the security and performance metrics and discussed results.

Addressing the research question Q2: To address the research question Q2, a proactive SDN-based MTD technique (named RHSM) is developed and presented in Chapter 4. RHSM hides both network and transport layer's attributes (e.g., IP address and port numbers) of the end-hosts in the SDN and provides end-to-end (e.,g., service-to-service) secure communication tunnel. It provides high network diversity by obfuscating the end-hosts' real identities, which significantly disturbs the attacker's scanning strategies and accordingly results in minimizing the attacker's success. Like FRVM, the architecture of RHSM and multiplexing (e.g., hosts and services) algorithms are developed for effective realizing of deploying it in SDNs. The performance of the RHSM is evaluated via simulations to analyze the effect of port address space scanned by attackers under varying the number of vulnerable services. Also, the overhead upon the operation of the RHSM in terms of the flow table size, the frequency of flow rules modifications, and the performance of an SDN controller is discussed.

Addressing the research question Q3: To address the research question Q3, a new suite of dynamic security metrics are developed and presented in Chapter 5. These security metrics are developed to capture the dynamic changes in a network over time, which are used to measure the effectiveness of the deployed MTD mechanism in SDNs. A variability represents the changing

pattern of the networks and/or systems' information (e.g., IP, ports, topology, paths, vulnerabilities). The metrics are grouped into three classes as they based on different kinds of security information and attack behaviors (e.g., network address, paths or topology, and attack stage). Network and/or system's configurations information (e.g., network state) is captured at different points of time and computed the variability for a scanning time window. T-HARM [47] is used to capture the path information of a network over time. Extensive simulations are performed and analyzed how the key parameters affect MTD based on the metrics and discussed the overall trends of results. Besides, the relationship between the metrics is discussed with the simulation results.

6.2 Limitations and Future Work

In this research, there are some limitations that can be investigated in future work in order to extend the research scope. In this section, we discuss the limitations of this research and the possible future work.

6.2.1 MTD for Emerging Network Technologies

Although all the developed MTD approaches (e.g., presented FRVM in Chapter 3, and RHSM in Chapter 4) and set of dynamic security metrics (e.g., presented in Chapter 5) of this thesis are typically based on SDN environments [136], the development of the MTD approaches can be further extended for securing the others emerging networked systems (e.g., Cloud, IoT, CPS). In this section, we discuss how the MTD approaches can be further developed in other emerging networked systems to increase the scope of the research.

MTD for Cloud Computing: The homogeneous computing infrastructures of cloud computing provides consistent and uniform quality services to their clients. However, maintaining the homogeneous computing infrastructure for uniform management of computing resources/services which

can lead the system more vulnerable [56]. MTD techniques can be explored to mitigate those security vulnerabilities or failure of the clouds. Various MTD approaches (e.g., shuffling of server replicas [80], VM migration [41, 122, 170], platform diversity / migration [24, 25], *etc.*) have been proposed for the clouds. However, the approaches are limited in changing configurations of system components, rather than shuffling of the network addresses or topology. Also, the MTD approaches (e.g., FRVM, RHSM) presented in this thesis are developed for the systems based on the SDNs. Therefore, it is important to investigate how new shuffling MTD approaches based on network addresses (e.g., IP address, ports) can be developed and deployed in the clouds. The newer cloud technologies (e.g., SDN-based Clouds [145]) provide effective deploying the MTD techniques in clouds strengthens not only security but also encourages innovation that leverages a new way of handling and managing dynamic configurations and infrastructure. With those technologies being leveraged, deploying the MTD techniques in clouds are more feasible for smaller and medium business houses as wellWang as big enterprises and government, which is highly scalable and cost-effective. However, it is inevitable to face challenges to make a good balance between security and performance in terms of minimizing security vulnerabilities and defense costs while maximizing service availability.

MTD for the Internet of Things (IoT): The advancement of IoT technologies have contributed to developing innovative applications in various domains, including cybersecurity [131]. However, due to its large scale and severe resource constraints relying on limited bandwidth and/or power, conventional security defense mechanisms (e.g., endpoint anti-virus software) has shown its limitations for applicability in IoT environments. Therefore, the threats or attacks encountered in IoT environments have been hurdles in providing seamless, normal services and operation of IoT-based systems [53]. The work presented in this thesis is typically based on the SDN-based enterprise network and did not consider any characteristics and constraints of the IoT

networks. Also, MTD mechanisms are emerging new technologies that can provide the capability to protect the IoT systems [26, 142, 167]). Thus, it is possible to extend our defense approaches presented in this thesis (Chapter 3 and 4) as proactive security defense mechanisms and for the IoT-based networks. Moreover, MTD makes the attackers harder to map the devices, exploit their vulnerabilities and launch the attacks. However, the constraints on IoT devices (e.g., CPU, energy consumption, memory) and network (e.g., low-bandwidth, high packet-loss) limit the effectiveness of MTD in IoT-based environments.

MTD for Cyber-Physical Systems (CPS): The CPS (e.g., automobiles, medical devices, smart grids) plays an important role in critical infrastructure, government, and everyday life [115]. The advance of CPS has been made along with increased cyber capabilities in terms of communications, networking, sensing, and computing as well as enhanced capabilities of physical systems with materials, hardware, and/or sensors/actuators [126]. The CPS also increases cybersecurity risks and attack surfaces, consequences of cyberattacks, which severe impact on human life. MTD approaches discussed in Chapters 3 and 4 of this thesis did not consider the characteristics of the CPS. Therefore, proactive security defense approaches are needed for strengthening the security and resilience of the CPS. Some MTD approaches (e.g., IP-hopping, MT6D) have been proposed to protect supervisory control and data acquisition (SCADA) system [121, 153], and smart gridsGroat et al. [59]. SCADA-based systems are generally used for monitoring and controlling physical devices span in large geographic distances, which are part of national infrastructures (e.g., water distribution, oil and natural gas pipelines, power grids, and transportation systems) and are critical to a nation's economy and safety. Leveraging the SDN technology for MTD approaches to SCADA systems can protect the CPS system with an additional layer of defenses. However, effective deployment of the MTD mechanisms should not adversely impact the core performance of a system, including safety, reliability, availability, and

predictability in running the operational services of the CPS.

6.2.2 Dynamic and Adaptable MTD Approaches

The state-of-the-art shuffling MTD techniques (e.g., [3], [74], [10], [99]) have been mostly used a fixed MTD interval time for changing the attack surfaces of the system. Although the MTD approaches presented in Chapter 3 and Chapter 4 of this thesis are proactive defense mechanisms, they still have used fixed MTD interval time since they change IP addresses and/or ports of the target system periodically with a certain fixed interval time. With a fixed MTD interval time, an attacker can easily learn how the attack surfaces (e.g., IP, ports, vulnerability) of the target system are being changed over time. Hence, investigating how MTD shuffling interval time can be made dynamic and adaptable to disrupt the chance of learning of the attacker and reduce the attack success could be a promising future research direction. The adaptable MTD shuffling strategy can be achieved based on the level of system vulnerabilities or attack patterns which requires the advanced detection mechanisms (e.g., IDS, IPS) or learning capability (e.g., Machine learning [49], Deep learning [134], Re-enforcement learning [171]) in a defending system.

6.2.3 Optimizing MTD Shuffling Frequency

MTD shuffling frequency (or MTD interval) significantly impacts on the security of the deployed MTD mechanism, which has been studied and discussed in Chapter 3 of this thesis. However, the optimization of the shuffling frequency is not discussed in this research. The optimal shuffling frequency determination is a key challenge issue in the current shuffling-based MTD researches [21]. Therefore, it is crucial to determine the optimal shuffling frequency that minimizes the cost of network hardening. Different approaches (e.g., Genetic-algorithm [160, 36, 38, 97], Machine-learning [134, 49, 146], and Game-theoretic [25, 135, 50, 161]) are widely used for modeling and

optimization of MTD problems.

GA-based MTD approaches can be used for identifying the optimal deployment of system configurations where evolutionary algorithms such as genetic algorithms (GAs) are often used to generate system configurations with high diversity for maximizing system security [177].

Machine-learning (ML)-based approaches can be used to hinder an attacker's learning or select the best defense strategy. Besides, the multi-objective reinforcement learning algorithm can be used to minimize the attack surface of a system [152]. ML-based MTD allows a system to capture evolving attack patterns with high scalability and applicability. However, as the performance of ML often requires a large amount of data for training to guarantee a certain level of prediction accuracy, then when there is a lack of data, the performance is less than desired even with high overhead and complexity.

Game theory is a mathematical model that allows modeling conflict and cooperation between two or more rational decision-makers (e.g., players). Game-theoretic approaches have been commonly used in modeling the interactions between an attacker and a defender in designing MTD techniques. Hence, in terms of the defender's perspective, the key goal of an MTD strategy is to identify a set of optimal system configuration policies to shift the attack surface, which can minimize risk and/or damage introduced by an action by the attacker [172]. On the other hand, the attacker aims to successfully launch its attack with minimum effort and maximum effectiveness in achieving its objectives. Hence, a two-player game well models the competition scenario between the attacker and the MTD-based defender. Different games (e.g., Markov [94], Bayesian Stackelberg [135], Stochastic [104]) can be formulated based on attacker behaviours and system models considered. The game-theoretic approaches can provide an effective way to consider each player's learning and, accordingly, their adaptive behavior to maximize their utility.

6.2.4 Advanced Attack Behaviors

We defined an attacker model and made assumptions for the attacker's capabilities and attack types considered in this research. We considered an external attacker with some knowledge toward the target system and the deployed defense mechanisms (e.g., IP and port address space range, MTDs) and repeatedly scanned the target system. However, real-world attacker can have different skills (e.g., adaptive [80]) and attack behaviors (e.g., APT [18], epidemic worms [178], DDoS [147]) which are not considered in our model. Also, as demonstrated in Section 2.3.3, most attack behaviors considered in the existing MTDs occur during the reconnaissance stage, which implies that MTD mainly deals with outside attackers as its primary goal is to protect a system before they break into the system. This significantly limits the applicability of MTD techniques, although MTD can enhance system security and performance by dealing with attackers beyond the reconnaissance stage (e.g., inside attackers). Therefore, proposed approaches of this thesis can be extended in the future considering a different type of attackers (e.g., insiders, adaptive attackers) and attack behaviors (e.g., APT, DDoS, epidemic attacks).

6.2.5 Performability Metrics

We developed a comprehensive set of dynamic security metrics for MTDs and presented in Chapter 5 of this thesis, which can be used to assess and evaluate the effectiveness of the deployed MTD mechanism in terms of security aspects of the networked systems. These metrics did not take performability (e.g., performance and availability) measurement into account. The impact of the deployed MTD on system performance and service availability (e.g., operational delay, connection drop) can be a critical design trade-off between security and performance. Thus, designing performability metrics (e.g., delay [110, 168], connection loss [23]) for measuring the effectiveness of MTD mechanisms in SDN could be a potential future research direction.

6.2.6 Validation in a real SDN-testbed

The proposed approaches have been validated via simulation experiments in a virtualized SDN environment using *Mininet* [91] emulator with a Ryu [133] controller. The simulation environment was set up as close as real network settings and experiments carried out with the use of the existing vulnerability database (e.g., NVD [113], but one of the limitations of this thesis is the lack of deployment in a real SDN-testbed for the validation. A limited number of existing SDN-based MTD work used a real SDN-testbed [103, 65, 158]. Real SDN-testbeds can help verify the performance of MTD techniques under more realistic system environments.

Chapter 7

Conclusions

The static configuration of the conventional networked systems provides asymmetric advantages to attackers so that they have enough time to study towards the target systems, identify vulnerabilities, and make a successful attack plan. MTD approaches have been evolved for creating the asymmetric uncertainty (i.e., asymmetric disadvantages) to the attackers by continuously and dynamically changing the networks or systems' configuration over time, therefore, harder to explore and predict. Also, flexibility and programmability features of software-defined networking facilitate to implement various cybersecurity network applications. There is a lack of proactive and dynamic defense mechanisms that dynamically reconfigure or change the networked systems' properties (e.g., IP, ports, topology) for protecting the software-defined networks. Similarly, there are no suitable security metrics that capture the network dynamics and measure the effectiveness of the deployed moving target defense mechanisms in the software-defined networks. This thesis identified three major problems in the existing network address shuffling-based MTD mechanisms and security metrics that can be used to measure the effectiveness of the MTD mechanisms deployed in software-defined networks. First, the existing network address shuffling MTD mechanisms are posing problems in the mapping of the IP addresses or port numbers of the networked systems. Secondly, current security defense mechanisms are unable to obfuscate

both hosts and vulnerable services of the target systems. Also, they are often deployed in conventional networked environments. Thirdly, the security metrics have been designed form the general system security perspective, so they are unable to capture and measure the dynamics of the networked systems introduced by the shuffling or reconfiguration event of the deployed MTD mechanisms.

So, in this research, following four distinct key contributions to knowledge in the development of moving target defense approaches and dynamic security metrics are made.

1. A new MTD approach has been developed for shuffling of the IP addresses of server machines and thwarting reconnaissance and scanning cyberattacks in the SDNs. A flexible IP shuffling mechanism called, FRVM was developed that provides an efficient method of mapping (i.e., multiplexing) or remapping (i.e., de-multiplexing) between virtual and real IP addresses of the servers, where each server machine can have multiple, random, time-variant IP addresses which creates an extremely diverse environment that significantly decreases attacker success. System architecture, address mapping algorithms, and communication protocols of the FRVM defense mechanism were developed. A proof-of-concept FRVM was implemented on SDN testbed (e.g., Mininet) in developing a separate SDN controller, named *FRVM controller*.

2. A comprehensive performance evaluation of the FRVM has been presented. A comparative study of the security and overhead of the FRVM with a typical static network counterpart was described using security metrics (e.g., attacker's scanning success probability, deterrence) and performance metrics (e.g., delay, throughput) respectively. The probabilistic analytical models were derived for both dynamic networks with FRVM and a typical static network without FRVM. The results were analyzed with the impact of key parameters (e.g., MTD interval time, no.

of vulnerable hosts, address space size) on security in terms of attacker success to scan and explore vulnerabilities in the target systems.

3. A proactive and dynamic SDN-based defense mechanism, RHSM has been developed for obfuscating both host's and services' identities and defending against the reconnaissance and scanning attacks in the SDN. The flexible address multiplexing concept of the FRVM defense mechanism has been adopted in the RHSM approach to map both IP addresses and port numbers of the end-hosts. The performance of the RHSM comparatively analyzed with a static counterpart network in terms of attack success probability with the existence of the different number of vulnerable services and overhead of flow update.

4. A new suite of dynamic security metrics for measuring the effectiveness of MTD has been developed. These metrics captured the dynamics of SDN-based networked systems' properties (e.g., IP address, ports, attack paths, topology) that are used to estimate the systems' variability at different time points. The developed metrics were categorized into three classes (e.g., network address based, attack paths based, and attack stages based) based on information they used and attack behaviors considered. Examples and simulation experiments were used to identify the key factors that significantly impact on the performance of the deploying MTD techniques in the SDNs.

References

[1] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha. Cyber deception: Virtual networks to defend insider reconnaissance. In *Proceedings of the 8th ACM CCS international workshop on managing insider security threats*, pages 57–68, 2016.

[2] S. Achleitner, T. L. Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha. Deceiving network reconnaissance using SDN-based virtual topologies. *IEEE Transactions on Network and Service Management*, 14:1098–1112, Dec. 2017.

[3] E. Al-Shaer, Q. Duan, and J. H. Jafarian. *Random Host Mutation for Moving Target Defense*, pages 310–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] H. Alavizadeh, D. S. Kim, J. B. Hong, and J. Jang-Jaccard. Effective security analysis for combinations of mtd techniques on cloud computing (short paper). In *Proceedings of the International Conference on Information Security Practice and Experience*, pages 539–548, 2017.

[5] H. Alavizadeh, J. B. Hong, J. Jang-Jaccard, and D. S. Kim. Comprehensive security assessment of combined MTD techniques for the cloud. In *Proceedings of the 5th ACM Workshop on Moving Target Defense (MTD)*, pages 11–20, 2018.

[6] H. Alavizadeh, J. Jang-Jaccard, and D. S. Kim. Evaluation for combination of shuffle and diversity on moving target defense strategy

for cloud computing. In *Proceedings of the 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*, pages 573–578, 2018.

[7] M. Albanese, A. D. Benedictis, S. Jajodia, and K. Sun. A moving target defense mechanism for MANETs based on identity virtualization. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pages 278–286, Oct. 2013.

[8] M. H. Almeshekah and E. H. Spafford. Cyber security deception. In *Cyber deception - Building the scientific foundation*, pages 23–50. Springer, 2016.

[9] N. Anderson, R. Mitchell, and I. R. Chen. Parameterizing moving target defenses. In *Proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–6, Nov. 2016.

[10] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. Defending Against Hitlist Worms using Network Address Space Randomization. In *Proc. the 2005 ACM Workshop on Rapid Malcode*, pages 30–40, 2005.

[11] F. Arnold, D. Guck, R. Kumar, and M. Stoelinga. Sequential and parallel attack tree modelling. In F. Koornneef and C. van Gulijk, editors, *Computer Safety, Reliability, and Security*, pages 291–299, 2015.

[12] A. Avizienis. The n-version approach to fault-tolerant software. *IEEE Transactions on Software Engineering*, 11(12):1491–1501, Dec. 1985.

[13] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman. Mitigating crossfire attacks using SDN-based moving target defense. In *Proceedings of the IEEE Conference on Local Computer Networks*, 2016.

[14] M. Azab, R. Hassan, and M. Eltoweissy. Chameleonsoft: a moving target defense system. In *Proceedings of the 7th International Conference*

on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pages 241–250, 2011.

[15] M. E. Baran and F. F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4(2):1401–1407, Apr. 1989.

[16] E. G. Barrantes, D. H. Ackley, S. Forrest, and D. Stefanović. Randomized instruction set emulation. *ACM Trans. Inf. Syst. Secur.*, 8(1):3–40, Feb. 2005.

[17] J. B. Bell and B. Whaley. *Cheating and deception*. Transaction Publishers, 1991.

[18] N. Ben-Asher, J. Morris-King, B. Thompson, and W. J. Glodek. Attacker skill defender strategies and the effectiveness of migration-based moving target defense in cyber systems. In *Proceedings of the 11th International Conference on Cyber Warfare and Security (ICCWS)*, 2016.

[19] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. ONOS: Towards an Open, Distributed SDN OS. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 1–6, 2014.

[20] J. W. Caddell. Deception 101-primer on deception. Technical report, DTIC Document, 2004.

[21] G. Cai, B. Wang, X. Wang, Y. Yuan, and S. Li. An introduction to network address shuffling. In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pages 185–190, Jan. 2016.

[22] G.-l. Cai, B.-s. Wang, W. Hu, and T.-z. Wang. Moving target defense:

State of the art and characteristics. *Frontiers of Information Technology & Electronic Engineering*, 17(11):1122–1153, Nov. 2016.

[23] T. E. Carroll, M. Crouse, E. W. Fulp, and K. S. Berenhaut. Analysis of Network Address Shuffling as a Moving Target Defense. In *2014 IEEE Int'l Conf. on Comm. (ICC)*, pages 701–706, Jun. 2014.

[24] K. M. Carter, H. Okhravi, and J. Riordan. Quantitative Analysis of Active Cyber Defenses Based on Temporal Platform Diversity. *arXiv preprint arXiv:1401.8255*, 2014.

[25] K. M. Carter, J. F. Riordan, and H. Okhravi. A game theoretic approach to strategy determination for dynamic platform defenses. In *Proceedings of the 1st ACM Workshop on Moving Target Defense (MTD)*, pages 21–30, 2014.

[26] V. Casola, A. D. Benedictis, and M. Albanese. A moving target defense approach for protecting resource-constrained distributed devices. In *Proceedings of the IEEE 14th International Conference on Information Reuse Integration (IRI)*, pages 22–29, 2013.

[27] L. Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *Digest of Papers FTCS-8: Eight Annual International Conference on Fault-Tolerant Computing*, pages 3–9, Toulouse, Jun. 1978.

[28] J. Cho, Y. Wang, I. Chen, K. S. Chan, and A. Swami. A survey on modeling and optimizing multi-objective systems. *IEEE Communications Surveys Tutorials*, 19(3):1867–1901, 2017.

[29] J.-H. Cho and N. Ben-Asher. Cyber defense in breadth: Modeling and analysis of integrated defense systems. *The Journal of Defense Modeling and Simulation*, 15(2):147–160, 2018.

[30] A. Chowdhary, S. Pisharody, and D. Huang. SDN Based Scalable MTD Solution in Cloud Network. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 27–36, 2016.

[31] A. Clark, K. Sun, and R. Poovendran. Effectiveness of IP address randomization in decoy-based moving target defense. In *Proceedings of the 52nd IEEE Conference on Decision and Control*, pages 678–685, Dec. 2013.

[32] A. Clark, K. Sun, L. Bushnell, and R. Poovendran. A game-theoretic approach to IP address randomization in decoy-based cyber defense. In *International Conference on Decision and Game Theory for Security*, pages 3–21, 2015.

[33] R. Colbaugh and K. Glass. Predictability-oriented defense against adaptive adversaries. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2721–2727, Oct. 2012.

[34] K. Compton and S. Hauck. Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys*, 34(2):171–210, Jun. 2002.

[35] W. Connell, D. A. Menasce, and M. Albanese. Performance Modeling of Moving Target Defenses with Reconfiguration Limits. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2018.

[36] M. Crouse and E. Fulp. A moving target environment for computer configurations using genetic algorithms. In *Proceeding of the 4th Symposium on Configuration Analytics and Automation (SAFECONFIG)*, pages 1–7, 2011.

[37] M. Crouse, B. Prosser, and E. W. Fulp. Probabilistic Performance Analysis of Moving Target and Deception Reconnaissance Defenses. In

Proceedings of the Second ACM Workshop on Moving Target Defense, MTD '15, pages 21–29, 2015.

[38] M. B. Crouse, E. W. Fulp, and D. A. Cañas. Improving the diversity defense of genetic algorithm-based moving target approaches. In *Proceedings of the National Symposium on Moving Target Research*, 2012.

[39] CVSS. NVD CVSS v2 Calculator, 2018. https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator.

[40] L. Daigle. WHOIS Protocol Specification. RFC 3912, Sep. 2004. URL https://rfc-editor.org/rfc/rfc3912.txt.

[41] B. Danev, R. Masti, G. Karame, and S. Capkun. Enabling secure vm-vtpm migration in private clouds. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC)*, pages 187–196, 2011.

[42] D. C. Daniel and K. L. Herbig. *Strategic military deception*. Pergamon, 1982.

[43] S. A. DeLoach, X. Ou, R. Zhuang, and S. Zhang. *Model-driven, moving-target defense for enterprise network security*, pages 137–161. Springer International Publishing, 2014.

[44] Q. Duan, E. Al-Shaer, and H. Jafarian. Efficient Random Route Mutation considering flow and network constraints. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 260–268, Oct. 2013.

[45] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront. MT6D: A Moving Target IPv6 Defense. In *2011 - MILCOM 2011 Military Communications Conference*, pages 1321–1326, Nov. 2011.

[46] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statist. Sci.*, 1(1):54–75, 02 1986.

[47] S. Y. Enoch, M. Ge, J. B. Hong, H. Alzaid, and D. S. Kim. A systematic evaluation of cybersecurity metrics for dynamic networks. *Computer Networks*, 144:216–229, 2018.

[48] D. Evans, A. Nguyen-Tuong, and J. Knight. Effectiveness of moving target defenses. In *Moving Target Defense*, pages 29–48. Springer, 2011.

[49] E. Farchi, O. Shehory, and G. Barash. Defending via strategic ML selection. *arXiv preprint arXiv:1904.00737*, 2019.

[50] X. Feng, Z. Zheng, D. Cansever, A. Swami, and P. Mohapatra. A signaling game model for moving target defense. In *Proceedings of the IEEE INFOCOM*, pages 1–9, 2017.

[51] M. Franz. E unibus pluram: massive-scale software diversity as a defense mechanism. In *Proceedings of the New Security Paradigms Workshop*, pages 7–16, 2010.

[52] L. Ge, W. Yu, D. Shen, G. Chen, K. Pham, E. Blasch, and C. Lu. Toward effectiveness and agility of network security situational awareness using moving target defense (MTD). In *Sensors and Systems for Space Applications VII*, volume 9085, pages 1–9, 2014.

[53] M. Ge, J. B. Hong, W. Guttmann, and D. S. Kim. A framework for automating security analysis of the internet of things. *Journal of Network and Computer Applications*, 83:12–27, Apr. 2017.

[54] M. Ge, J. Cho, C. Kamhoua, and D. S. Kim. Optimal deployments of defense mechanisms for internet of things. In *Proceedings of the International Workshop on Secure Internet of Things (SIoT)*, Sep. 2018.

[55] A. Gherbi and R. Charpentier. Diversity-based approaches to software systems security. In *Proceedings of the International Conference on Security Technology*, pages 228–237, 2011.

[56] A. Ghosh and I. Arce. Guest editors' introduction: In cloud computing we trust - but should we? *IEEE Security & Privacy*, 8(6):14–16, Nov. 2010.

[57] A. K. Ghosh, D. Pendarakis, and W. H. Sanders. National Cyber Leap Year Summit 2009 Co-Chairs' Report. Technical report, Federal Networking and Information Technology Research and Development (NITRD) Program, Sep. 2009.

[58] M. Green, D. C. MacFarland, D. R. Smestad, and C. A. Shue. Characterizing network-based moving target defenses. In *Proceedings of the 2nd ACM Workshop on Moving Target Defense (MTD)*, pages 31–35, 2015.

[59] S. Groat, M. Dunlop, W. Urbanksi, R. Marchany, and J. Tront. Using an IPv6 moving target defense to protect the smart grid. In *IEEE PES Innovative Smart Grid Technologies (ISGT)*, pages 1–7, Jan. 2012.

[60] Y. Han, W. Lu, and S. Xu. Characterizing the power of moving target defense via cyber epidemic dynamics. In *Proceedings of the Symposium and Bootcamp on the Science of Security*, volume 10, pages 1–12, 2014.

[61] E. Hernandez-Valencia, S. Izzo, and B. Polonsky. How will nfv/sdn transform service provider opex? *IEEE Network*, 29(3):60–67, May 2015. ISSN 1558-156X.

[62] V. Heydari, S. Kim, and S. Yoo. Scalable Anti-Censorship Framework Using Moving Target Defense for Web Servers. *IEEE Transactions on Information Forensics and Security*, 12(5):1113–1124, May 2017.

[63] J. Hong and D. Kim. Scalable Security Analysis in Hierarchical Attack Representation Model using Centrality Measures. In *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW)*, pages 1–8, 2013.

[64] J. B. Hong and D. S. Kim. Assessing the Effectiveness of Moving Target Defenses Using Security Models. *IEEE Trans. Dependable and Secure Comput.*, 13(2):163–177, 2016.

[65] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim. Optimal Network Reconfiguration for Software Defined Networks using Shuffle-based Online MTD. In *IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2017.

[66] J. B. Hong, S. Y. Enoch, D. S. Kim, A. Nhlabatsi, N. Fetais, and K. M. Khan. Dynamic security metrics for measuring the effectiveness of moving target defense techniques. *Computers Security*, 79:33 – 52, 2018.

[67] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra. Dynamic defense strategy against advanced persistent threat with insiders. In *Proceedings of the IEEE INFOCOM*, pages 747–755, 2015.

[68] Y. Huang and A. K. Ghosh. Introducing diversity and uncertainty to create moving attack surfaces for web services. In *Moving Target Defense*, pages 131–151. Springer, 2011.

[69] Y. Huang, A. K. Ghosh, T. Bracewell, and B. Mastropietro. A security evaluation of a novel resilient web serving architecture: Lessons learned through industry/academia collaboration. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 188–193, 2010.

[70] E. M. Hutchins, M. J. Cloppert, and R. M. Amin. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *Leading Issues in Information Warfare & Security Research*, 2011.

[71] T. Jackson, B. Salamat, A. Homescu, K. Manivannan, G. Wagner, A. Gal,

S. Brunthaler, C. Wimmer, and M. Franz. Compiler-generated software diversity. In *Moving Target Defense*, pages 77–98. Springer, 2011.

[72] T. Jackson, A. Homescu, S. Crane, P. Larsen, S. Brunthaler, and M. Franz. *Diversifying the Software Stack Using Randomized NOP Insertion*, pages 151–173. Springer New York, 2013.

[73] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In *Proc. 1st Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 127–132, 2012.

[74] J. H. Jafarian, E. Al-Shaer, and Q. Duan. An Effective Address Mutation Approach for Disrupting Reconnaissance Attacks. *IEEE Trans. Information Forensics and Security*, 10(12):2562–2577, Dec. 2015.

[75] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Adversary-aware IP Address Randomization for Proactive Agility Against Sophisticated Attackers. In *2015 IEEE Conf. Compu. Comm. (INFOCOM)*, pages 738–746, Apr. 2015.

[76] J. H. H. Jafarian, E. Al-Shaer, and Q. Duan. Spatio-temporal Address Mutation for Proactive Cyber Agility Against Sophisticated Attackers. In *Proc. 1st ACM Workshop on Moving Target Defense*, pages 69–78, 2014.

[77] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[78] A. Jangda, M. Mishra, and B. De Sutter. Adaptive just-in-time code diversification. In *Proceedings of the 2nd ACM Workshop on Moving Target Defense (MTD)*, pages 49–53, 2015.

[79] Q. Jia, K. Sun, and A. Stavrou. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. *22nd Int'l Conf. on Computer Comm. and Networks (ICCCN)*, pages 1–9, 2013.

[80] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell. Catch Me If You Can: A Cloud-Enabled DDoS Defense. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 264–275, 2014.

[81] P. Kampanakis, H. Perros, and T. Beyene. SDN-based solutions for Moving Target Defense network protection. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, Jun. 2014.

[82] M. Karakus and A. Durresi. Economic viability of software defined networking (sdn). *Computer Networks*, 135:81 – 95, 2018. ISSN 1389-1286.

[83] G. S. Kc, A. D. Keromytis, and V. Prevelakis. Countering Code-injection Attacks with Instruction-set Randomization. In *Proc. 10th ACM Conf. on Computer and Comm. Security (CCS)*, pages 272–280, 2003.

[84] A. D. Keromytis, R. Geambasu, S. Sethumadhavan, S. J. Stolfo, J. Yang, A. Benameur, M. Dacier, M. Elder, D. Kienzle, and A. Stavrou. The MEERKATS cloud security architecture. In *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops*, Jun. 2012.

[85] D. Kewley, R. Fink, J. Lowry, and M. Dean. Dynamic approaches to thwart adversary intelligence gathering. In *DISCEX '01. Proc. DARPA Information Survivability Conf. amp; Exposition II*, volume 1, pages 176–185, 2001.

[86] C. Kil, J. Jun, C. Bookholt, J. Xu, and P. Ning. Address Space Layout Permutation (ASLP): Towards Fine-Grained Randomization of Commodity Software. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, ACSAC '06, pages 339–348, 2006.

[87] J. Knight, J. Davidson, A. Nguyen-Tuong, J. Hiser, and M. Co. Diversity in cybersecurity. *Computer*, 49(4):94–98, Apr. 2016.

[88] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE*, 103(1):14–76, Jan. 2015.

[89] M. Kuźniar, P. Perešíni, and D. Kostić. What you need to know about sdn flow tables. In J. Mirkovic and Y. Liu, editors, *Passive and Active Measurement*, pages 347–359, 2015.

[90] J. H. Lala and F. B. Schneider. It monoculture security risks and defenses. *IEEE Security Privacy*, 7(1):12–13, Jan. 2009.

[91] B. Lantz, B. Heller, and N. McKeown. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, 2010.

[92] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz. SoK: Automated software diversity. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 276–291, 2014.

[93] H. C. J. Lee and V. L. L. Thing. Port Hopping for Resilient Networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, volume 5, pages 3291–3295 Vol. 5, Sep. 2004.

[94] C. Lei, D. H. Ma, and H. Q. Zhang. Optimal Strategy Selection for Moving Target Defense Based on Markov Game. *IEEE Access*, 5:156–169, 2017.

[95] Y. Li, R. Dai, and J. Zhang. Morphing communications of cyber-physical systems towards moving-target defense. In *2014 IEEE International Conference on Communications (ICC)*, pages 592–598, June 2014.

[96] C. Ltd. *Ubuntu Operating System, Version 17.10*. London, United Kingdom, 2018. URL https://www.ubuntu.com/.

[97] B. Lucas, E. W. Fulp, D. J. John, and D. Cañas. An initial framework for evolving computer configurations as a moving target defense. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR)*, pages 69–72, 2014.

[98] Y.-B. Luo, B.-S. Wang, and G.-L. Cai. Effectiveness of port hopping as a moving target defense. In *Proceedings of the 7th International Conference on Security Technology (SecTech)*, pages 7–10, 2014.

[99] Y. B. Luo, B. S. Wang, X. F. Wang, X. F. Hu, G. L. Cai, and H. Sun. RPAH: Random Port and Address Hopping for Thwarting Internal and External Adversaries. In *In Proc. 2015 IEEE Trustcom/BigDataSE/ISPA*, pages 263–270, 2015.

[100] Y.-b. Luo, B.-s. Wang, X.-f. Wang, and B.-f. Zhang. A keyed-hashing based self-synchronization mechanism for port address hopping communication. *Frontiers of Information Technology & Electronic Engineering*, 18(5):719–728, May 2017.

[101] G. F. Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, USA, 2009.

[102] D. Ma, L. Wang, C. Lei, Z. Xu, H. Zhang, and M. Li. POSTER: Quantitative Security Assessment Method Based on Entropy for Moving Target Defense. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 920–922, 2017.

[103] D. C. MacFarland and C. A. Shue. The SDN Shuffle: Creating a Moving-Target Defense Using Host-based Software-Defined Networking. In *Proc. 2nd ACM Workshop on Moving Target Defense*, pages 37–41, 2015.

[104] P. K. Manadhata. Game Theoretic Approaches to Attack Surface Shifting. In *Moving Target Defense II*, pages 1–13. Springer, 2013.

[105] P. K. Manadhata and J. M. Wing. *A Formal Model for a System's Attack Surface*, pages 1–28. Springer, New York, 2011.

[106] P. K. Manadhata and J. M. Wing. An Attack Surface Metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, May 2011.

[107] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev.*, 38(2): 69–74, Mar. 2008.

[108] J. Medved, R. Varga, A. Tkacik, and K. Gray. OpenDaylight: Towards a Model-driven SDN Controller Architecture. In *IEEE 15th Int'l Sym. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, 2014.

[109] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev. NetHide: secure and practical network topology obfuscation. In *Proceedings of the 27th USENIX Security Symposium*, pages 693–709, 2018.

[110] C. Morrell, J. S. Ransbottom, R. Marchany, and J. G. Tront. Scaling IPv6 address bindings in support of a moving target defense. In *Proceedings of the 9th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 440–445, Dec. 2014.

[111] Nessus. Nessus Professional, 2018. https://www.tenable.com/products/nessus/nessus-professional.

[112] S. Neti, A. Somayaji, and M. E. Locasto. Software diversity: Security, entropy and game theory. In *Proceedings of the USENIX Summit on Hot Topics in Security*, 2012.

[113] NIST. National Vulnerability Database, 2019. https://nvd.nist.gov/.

[114] D. of Homeland Security. Moving target defense, 2018. URL https://www.dhs.gov/science-and-technology/csd-mtd.

[115] D. of Homeland Security. Cyber physical systems security, 2019. URL https://www.dhs.gov/science-and-technology/cpssec.

[116] H. Okhravi, A. Comella, E. Robinson, and J. Haines. Creating a cyber moving target for critical infrastructure applications using platform diversity. *International Journal of Critical Infrastructure Protection*, 5 (1):30–39, 2012.

[117] H. Okhravi, M. Rabe, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein. Survey of cyber moving targets. Technical Report 1166, Massachusetts Inst of Technology Lexington Lincoln Lab, 2013.

[118] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. Finding focus in the blur of moving-target techniques. *IEEE Security & Privacy*, 12(2): 16–26, 2014.

[119] H. Okhravi, W. W. Streilein, and K. S. Bauer. Moving target techniques: Leveraging uncertainty for cyber defense. In *Lincoln Laboratory Journal*, volume 22, pages 100–109, 2016.

[120] Oracle. *VM VirtualBox Software, Version 5.2.12*, 2018. URL https://www.virtualbox.org/.

[121] A. C. Pappa, A. Ashok, and M. Govindarasu. Moving target defense for securing smart grid communications: Architecture, implementation, and

evaluation. In *Proceedings of the IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, Apr. 2017.

[122] W. Peng, F. Li, C.-T. Huang, and X. Zou. A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In *IEEE International Conference on Communications (ICC)*, pages 804–809, 2014.

[123] T. Penner and M. Guirguis. Combating the Bandits in the Cloud: A moving target defense approach. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 411–420, 2017.

[124] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending networking into the virtualization layer. In *Hotnets*, 2009.

[125] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, et al. The Design and Implementation of Open vSwitch. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 117–130, 2015.

[126] R. Poovendran. Cyber–Physical Systems: Close encounters between two parallel worlds [point of view]. *Proceedings of the IEEE*, 98(8):1363–1366, Aug. 2010.

[127] G. Portokalidis and A. D. Keromytis. *Global ISR: Toward a Comprehensive Defense Against Unauthorized Code Execution*, pages 49–76. Springer New York, 2011.

[128] A. Prakash and M. P. Wellman. Empirical game-theoretic analysis for moving target defense. In *Proceedings of the 2nd ACM Workshop on Moving Target Defense (MTD)*, pages 57–65, 2015.

[129] M. A. Rahman, M. H. Manshaei, and E. Al-Shaer. A game-theoretic approach for deceiving remote operating system fingerprinting. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pages 73–81, 2013.

[130] M. A. Rahman, E. Al-Shaer, and R. B. Bobba. Moving target defense for hardening the security of the power system state estimation. In *Proceedings of the 1st ACM Workshop on Moving Target Defense (MTD)*, pages 59–68, 2014.

[131] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed Internet-of-Things. *Computer Networks*, 57(10):2266–2279, 2013.

[132] J. Rowe, K. N. Levitt, T. Demir, and R. Erbacher. Artificial diversity as maneuvers in a control theoretic moving target defense. In *National Symposium on Moving Target Research*, 2012.

[133] Ryu project team. *Ryu SDN Framework*. Ryu SDN Framework Community, 2018. URL https://osrg.github.io/ryu-book/en/Ryubook.pdf.

[134] S. Sengupta, T. Chakraborti, and S. Kambhampati. MTDeep–Boosting the Security of Deep Neural Nets Against Adversarial Attacks with Moving Target Defense. In *Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Engineering Dependable and Secure Machine Learning Systems*, 2017.

[135] S. Sengupta, S. G. Vadlamudi, S. Kambhampati, A. Doupé, Z. Zhao, M. Taguinod, and G.-J. Ahn. A game theoretic approach to strategy generation for moving target defense in web applications. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 178–186, 2017.

[136] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Comm. Magaz.*, 51(7):36–43, Jul. 2013.

[137] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the Effectiveness of Address-space Randomization. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 298–307, 2004.

[138] Z. Shan, I. Neamtiu, Z. Qian, and D. Torrieri. Proactive restart as cyber maneuver for Android. In *IEEE Military Communications Conference*, pages 19–24, 2015.

[139] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J. Cho, and T. J. Moore. FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks. In *Proc. 17th IEEE Int'l Conf. Trust, Security And Privacy (TrustCom)*, pages 579–587, Aug. 2018.

[140] D. P. Sharma, J. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim. Random Host and Service Multiplexing for Moving Target Defense in Software-Defined Networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2019.

[141] W. L. Sharp. Military Deception. Technical report, Department of Defense, 2006. Joint Publication 3-13.4.

[142] M. Sherburne, R. Marchany, and J. Tront. Implementing Moving Target IPv6 Defense to Secure 6LoWPAN in the Internet of Things and Smart Grid. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR)*, pages 37–40, 2014.

[143] L. Shi, C. Jia, S. Lü, and Z. Liu. *Port and Address Hopping for*

Active Cyber-Defense, pages 295–300. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[144] R. Skowyra, K. Bauer, V. Dedhia, and H. Okhravi. Have No PHEAR: Networks Without Identifiers. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 3–14, 2016.

[145] J. Son and R. Buyya. A Taxonomy of Software-Defined Networking (SDN)-Enabled Cloud Computing. *ACM Comput. Surv.*, 51(3):59:1–59:36, May 2018.

[146] Q. Song, Z. Yan, and R. Tan. Moving Target Defense for Deep Visual Sensing against Adversarial Examples. *CoRR*, abs/1905.13148, 2019.

[147] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo. DDoS Defense using MTD and SDN. In *IEEE/IFIP Network Operations and Management Symposium*, 2018.

[148] M. Taguinod, A. Doupé, Z. Zhao, and G.-J. Ahn. Toward a moving target defense for web applications. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 510–517, 2015.

[149] C. Taylor and J. Alves-Foss. Diversity as a computer defense mechanism. In *Proceedings of the Workshop on New Security Paradigms*, pages 11–14, 2005.

[150] J. Taylor, K. Zaffarano, B. Koller, C. Bancroft, and J. Syversen. Automated Effectiveness Evaluation of Moving Target Defenses: Metrics for Missions and Attacks. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 129–134, 2016.

[151] M. Thompson, N. Evans, and V. Kisekka. Multiple OS Rotational Environment an Implemented Moving Target Defense. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pages 1–6, 2014.

[152] B. Tozer, T. Mazzuchi, and S. Sarkani. Optimizing attack surface and configuration diversity using multi-objective reinforcement learning. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 144–149, Dec. 2015.

[153] J. Ulrich, J. Drahos, and M. Govindarasu. A symmetric address translation approach for a network layer moving target defense to secure power grid networks. In *2017 Resilience Week (RWS)*, pages 163–169, Sep. 2017.

[154] M. Van Gundy and H. Chen. Noncespaces: Using randomization to enforce information flow tracking and thwart cross-site scripting attacks. In *Network and Distributed System Security Symposium (NDSS)*, 2009.

[155] S. Vikram, C. Yang, and G. Gu. Nomad: Towards non-intrusive moving-target defense against web bots. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pages 55–63, 2013.

[156] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou. A moving target DDoS defense mechanism. *Computer Communications*, 46:10–21, 2014.

[157] J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, and H. Zhang. CHAOS: an SDN-based Moving Target Defense System. *CoRR*, abs/1704.01482, 2017.

[158] Y. Wang, Q. Chen, J. Yi, and J. Guo. U-TRI: Unlinkability Through Random Identifier for SDN Network. In *Proceedings of the 2017 Workshop on Moving Target Defense*, MTD '17, pages 3–15, 2017.

[159] B. Ward, S. Gomez, R. Skowyra, D. Bigelow, J. Martin, J. Landry, and H. Okhravi. Survey of cyber moving targets. Technical Report 1228, Massachusetts Inst of Technology Lexington Lincoln Lab, Jan. 2018.

[160] M. L. Winterrose and K. M. Carter. Strategic Evolution of Adversaries Against Temporal Platform Diversity Active Cyber Defenses. In *Proceedings of the 2014 Symposium on Agent Directed Simulation*, ADS '14, pages 9:1–9:9, 2014.

[161] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman. Moving Target Defense Against DDoS Attacks: An Empirical Game-Theoretic Analysis. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 93–104, 2016.

[162] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman. Moving target defense against DDoS attacks: An empirical game-theoretic analysis. In *Proceedings of the 3rd ACM Workshop on Moving Target Defense (MTD)*, pages 93–104, 2016.

[163] J. Xu, Z. Kalbarczyk, and R. K. Iyer. Transparent runtime randomization for security. In *22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings.*, pages 260–269, Oct 2003.

[164] J. Yackoski, P. Xie, H. Bullen, J. Li, and K. Sun. A self-shielding dynamic network architecture. In *2011 - MILCOM 2011 Military Communications Conference*, pages 1381–1386, Nov 2011.

[165] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari. Architecture-based self-protecting software systems. In *Proceedings of the 9th international ACM SIGSOFT Conference on Quality of Software Architectures*, pages 33–42, 2013.

[166] K. Zaffarano, J. Taylor, and S. Hamilton. A Quantitative Framework for Moving Target Defense Effectiveness Evaluation. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, pages 3–10, 2015.

[167] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront. Changing the game: A micro moving target IPv6 defense for the internet of things. *IEEE Wireless Communications Letters*, 7(4):578–581, Aug. 2018.

[168] H. Zhang, C. Lei, D.-X. Chang, and Y. jie Yang. Network moving target defense technique based on collaborative mutation. *Computers & Security*, 70:51–71, 2017.

[169] L. Zhang, Y. Guo, H. Yuwen, and Y. Wang. A Port Hopping Based DoS Mitigation Scheme in SDN Network. In *2016 12th International Conference on Computational Intelligence and Security (CIS)*, pages 314–317, Dec 2016.

[170] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang. Incentive compatible moving target defense against VM-colocation attacks in clouds. In *Proceedings of the IFIP International Information Security Conference*, pages 388–399, 2012.

[171] M. Zhu, Z. Hu, and P. Liu. Reinforcement learning algorithms for adaptive cyber defense against Heartbleed. In *Proceedings of the 1st ACM Workshop on Moving Target Defense (MTD)*, pages 51–58, 2014.

[172] Q. Zhu and T. Başar. Game-theoretic approach to feedback-driven multi-stage moving target defense. In *Proceedings of the International Conference on Decision and Game Theory for Security*, pages 246–263, 2013.

[173] Q. Zhu, A. Clark, R. Poovendran, and T. Başar. Deceptive routing games. In *Proceedings of the IEEE 51st Annual Conference on Decision and Control (CDC)*, pages 2704–2711, 2012.

[174] R. Zhuang, S. Zhang, S. DeLoach, X. Ou, and A. Singhal. Simulation-based Approaches to Studying Effectiveness of Moving-Target Network

Defense. In *Proceedings of National Symposium on Moving Target Research*, 2012.

[175] R. Zhuang, S. Zhang, A. Bardas, S. DeLoach, X. Ou, and A. Singhal. Investigating the application of moving target defenses to network security. In *Proceeding of the 6th International Symposium on Resilient Control Systems (ISRCS)*, pages 162–169, 2013.

[176] R. Zhuang, S. A. DeLoach, and X. Ou. A model for analyzing the effect of moving target defenses on enterprise networks. In *Proceedings of the 9th ACM Annual Cyber and Information Security Research Conference*, pages 73–76, 2014.

[177] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a Theory of Moving Target Defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, MTD '14, pages 31–40, 2014.

[178] C. C. Zou, D. Towsley, and W. Gong. On the Performance of Internet Worm Scanning Strategies. *Perform. Eval.*, 63(7):700–723, Jul. 2006.

Appendix A

Related Publications

All the work presented in this thesis has been published or submitted for publication in the peer-reviewed journals and conferences, as listed in the following.

1. **D. P. Sharma**, D. S. Kim, S. Yoon, H. Lim, J. Cho and T. J. Moore. FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks. In *Proceedings of 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 579-587, Aug. 2018.

2. **D. P. Sharma**, J. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim. Random Host and Service Multiplexing for Moving Target Defense in Software-Defined Networks. In *Proceedings of ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2019.

3. **D. P. Sharma**, S. Y. Enoch, J. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim. Dynamic Security Metrics for Software-Defined Network-based Moving Target Defense. Submitted to *24th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2019)*, Dec. 1-3, 2019, Kyoto, Japan.

4. J. Cho, **D. P. Sharma**, H. Alavizadeh, N. Ben-Asher, S. Yoon, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense," In *IEEE Communications Surveys & Tutorials*, Volume 22, No.1, Pages 709-745, Firstquarter 2020..

5. C. Dishington, **D. P. Sharma**, D. S. Kim, J. Cho, T. J. Moore, F. F. Nelson. Security and Performance Assessment of IP Multiplexing Moving Target Defense in Software Defined Networks. In *Proceedings of 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, 5-8 Aug. 2019,

6. **D. P. Sharma**, C. Dishington, J. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim. Flexible Address Multiplexing for Moving Target Defense in Software-Defined Networks. (To be submitted in *IEEE Transaction on Dependability and Security Computing*)